Issues in Epistemic and Modal Logics and their applications

A dissertation submitted to the
Faculty of Philosophy of the University of Tübingen
in partial fulfillment of the requirements
for the degree of Doctor of Philosophy

by

Cezar A. Mortari

from

Santa Maria (Brazil)

1991

Hauptberichterstatter: Prof. Dr. Dr. Walter Hoering

Mitberichterstatter: Prof. Dr. Peter Schroeder-Heister

Dekan: Prof. Dr. Günter Figal

Tag der mündlichen Prüfung: 28.1.1991

Gedruckt mit Genehmigung der Philosophischen Fakultät der Universität Tübingen

This work has been done also thanks to a scholarship provided by CAPES, Brazil.

For Daniela,
in love.

Acknowledgements

Many people contributed in several ways to make this work possible, so any tentative to write a list would certainly end up being unjust by leaving someone out. To all of you, my warm appreciation.

To some people, however, I owe a special debt:

Many thanks to Prof. Walter Hoering, my adviser *de jure*, who was always there, ready to help, when I needed him, and who has done his best to keep my way free of troubles.

This work would not have been accomplished at all without the constant support and encouragement of Prof. Franz Guenthner, my *de facto* adviser, who, besides showing me a lot of the logical landscape, also introduced me to computers and to the joys of programming. As the former director of the Seminar für Natürlich-Sprachliche Systeme (SNS) of the University of Tübingen, Prof. Guenthner provided me with a wonderful working and learning environment, which I will never forget. Thank you very, very much!

I would like also to thank all my colleagues and friends, both at the SNS and at the Federal University of Santa Catarina (UFSC), Brazil, for the help along the way. In particular, my thanks to Prof. Sônia Felipe.

Finally, my deepest gratitude to Daniela, for her love, patience and understanding.

Contents

0

Introduction & Road Map

Mihi a docto doctore
domandatur causam et rationem quare
opium facit dormire?
A quoi respondeo
Quia est in eo
Virtus dormitiva,
Cujus est natura
Sensus assoupire.

MOLIÈRE, *Le Malade Imaginaire.*

0.1 Painting the background

The title of this work, *Issues in epistemic and modal logics and their applications*, is obviously a very comprehensive one, the reason for this being the fact that the contents reflect my multiple interests during the time I have been studying and working in Tübingen. Such a title doesn't tell us very much about exactly what the contents are, or which issues are actually going to be considered, so of course I'll have to say a few more words introducing the work and narrowing its subject matter. However, before we get down to discussing the specific research problems, I guess it would be nice, and even necessary, to dwell awhile on some preliminaries describing the big mosaic of which this work is hopefully going to be a small piece. This surely will give the reader a better understanding of what I'm up to here, and why.

To begin with, one could ask, why (one more work in) modal logics? Such a question is perhaps to be expected since probably everybody has, at least once, heard about these logics, and, if they are not one's working area, one probably has this idea that modal logics only deal with funny concepts like necessity and possibility and contingency; in other words, that they deal with a lot of pretty metaphysical stuff—just remember all that talking about Leibnizian "possible worlds" (of which ours is supposed to be the best one), and worlds being "accessible", and "parallel universes", and so on, until one is caught discussing how many possible fat men stand on that doorway.[1] One would hardly suspect that modal logics could be of use in this (possible) world and utilitarian times of ours.

Now, to tell the plain truth, the interests of modal logics do *not*—at least *not only*—concern metaphysical talking about possible worlds. Modality is in fact a very broad notion, and considerations about necessity and possibility deal with just one small side of it, namely with what is usually called *alethic modality*. ("Alethic" comes from the Greek word for "truth".) Necessity and possibility are said to be "modes of truth"; i.e., they refer to the way in which a proposition can be true, like "necessarily true", "possibly true", "impossibly true" (that is, "necessarily false"), and so on.

Now alethic modal logic was the first one to get developed: we can trace its beginning down to the beginning of logic itself, namely to Aristotle. In his works *De Interpretatione* and *Analytica Priora* he discusses logical interconnections between modal notions—such as necessary, impossible, possible, and permitted—as well as giving some thought to the theory of modal syllogisms, that is, syllogisms which have modalized premisses and conclusion (cf. [Lem77], p. 1–2). (As an example, "all animals are necessarily mortal" and "all humans are necessarily animals", *ergo* "all humans are necessarily mortal".) According to Lemmon, much of Aristotle's discussion is quite confused, but "[its] outcome is a remarkably correct set of implications" ([Lem77], p. 1).

From Aristotle's time on, beginning with his own school, not forgetting the Stoics and Megarians, and going until the end of the Middle Ages, there were a lot of people interested in and working with modal notions, with sometimes rather interesting contributions. We could mention, as an illustration, Diodorus Chronus, who gave definitions of necessity and possibility by means of temporal notions ("the possible is that which either is or will be"); or the medieval discussion about *de dicto* and *de re* modalities; or Pseudo-Scotus, who studied, besides "necessary" and "possible", other modalities such as "it is known that" or "it is believed that", thus antecipating epistemic logics (cf. [Lem77, p. 4]). Afterwards, however, not very much happened, the interregnum between the end of the Dark Ages and the nineteenth century not being the best possible modal logical world. Thus the modern development of modal logic starts only in this century with the work of C.I. Lewis, whose main contribution, one could say, were the so-called "Lewis systems", S1–S5, which axiomatize increasingly strong conceptions of necessity.[2]

To make it short, thanks to alethic modal logic's early, aristotelic beginning, the term "modal" got stuck with this member of the logic family—it was the only one around in town. But as the years went by, alethic's younger sisters came into existence and grew into logics in their own right, and it became then usual to employ the expression "modal logic" in a broader sense, which wasn't only restricted to modes of truth. Thus, today, we classify as modal logics, beyond the alethic ones, also *temporal logics*, *deontic logics*, *epistemic logics*, and so forth. In a sense, one could label as "modal logics" all logical systems in which one extends the language of classical logic by means of adding a certain kind of new operators, the so called *intensional* ones. Intensional operators are those which are not truth-functions of the propositions to which they apply. For instance:

[1] (Cf. [Qu80], p. 4) By a very suspicious coincidence, their number is exactly the same as the number of angels in the eye of a needle—or was it in the *head* of a *pin*...?

[2] To tell the truth, Lewis' main interest was not formalizing several notions of necessity and possibility; he was actually working on different conceptions of implication, trying to avoid the paradoxes of material implication. In the course of his investigations he arrived to the *strict* implication, which one can characterize as the necessity of the conditional—this is where necessity comes into picture. By the way, readers wanting to know more about the historical development of modal logic are referred to [KK62, Lem77, HC72], where additional bibliography can also be found.

"It is necessary that ..."
"It is possible that ..."
"It will be the case that ..."
"Darth Vader believes that ..."
"It is obligatory that ..."

In the Customary Way Of Doing Things, one takes the classical logic, say the classical propositional logic, and adds to its language two new operators, '□' (*box*) and '◇' (*diamond*), also introducing some axioms and inference rules involving them. Usually the box gets interpreted as "necessarily", and the diamond as "possibly". But someone can choose to say that '$\Box p$' means "always p" (where p stands for some proposition), so he's doing *tense* logic. And someone else takes '$\Box p$' to mean "Yoda knows that p", so she's doing *epistemic* logic. Thus one could venture that only the way you interpret the box (necessary, knows, always, provable) and the diamond will give a cue about which kind of logic you are doing. Of course, depending on the different interpretations of the operators, different formulas can or will hold, or not, but very often the same calculus is said to be both the nicest alethic and the nicest epistemic logic, for instance.

As to the possible worlds we mentioned above, we are going to find them in the so-called *possible-world semantics* for modal logics. In the case of classical logic, e.g. in a semantics for the propositional calculus, to evaluate a formula we proceed by looking at one model (typically a function assigning truth-values to propositional variables) and then computing this formula's value. In the case of modal logics, we have to consider more "models" at the same time. If we understand a model to be a kind of "world description", this amounts to say that in the modal logic case more worlds have to come into the picture. Thus a proposition is *necessary* (in some world) not only if it holds with respect to this world, but also if it holds in every other possible world (or, at least, in every other possible world which is *accessible* to the one we're in). And a proposition is *possible* if it holds in some (accessible) world. As one can see, this kind of semantics matches well the old Leibnizian account of necessity and possibility.

Before going on, let me remark that the above characterization of modal logics—as extensions of the classical one—is obviously too restrictive. Actually it just applies to what one could label *classical modal logic* (see [BS84]). According to this view, modal logics do not try to *substitute* the classical one, just *extend* it and make it more powerful. But one could as well take another position, choosing as underlying logic a rival of the classical one: intuitionistic logic, for instance, or relevance logic. If we now extend it by adding modal operators, we'll end up with, say, paraconsistent modal logics, or relevant modal logics, and so on. (For relevant modal logics, see e.g. [AB75, Fu88].)

Thus we have seen that there are many other possibilities besides plain alethic modal logic, so with yet another work in modal logics one won't necessarily end up being a metaphysician.[3] Nevertheless, before we jump to the conclusion that alethic modal logics are *prima facie* metaphysical and hence uninteresting, let me remind you that this is *absolutely not* the case: there are also several other interpretations of "necessity" to choose from. One can of course talk about a metaphysical kind of necessity, concerning possible worlds, but "necessarily" can also mean "according to the laws of physics"

[3] There is of course nothing *wrong* in doing Metaphysics, but this word is often used as an accusation, thus...

or "after the program terminates", or "according to my beliefs" (cf. [FV85], p. 2; also [Go87], p. 6). We can even talk about "historical necessity", for that matter.

Having thus learned from these general remarks what modal logics are, let us talk a little bit about their importance. To begin with, surely in philosophy:

> Some of the problems raised by modal logic seem to us to be among the most important and fundamental in philosophy, but it would require a separate book, and a very different one from ours, to discuss them adequately. In our view there is also a link of a different kind between philosophy and modal logic, in that modal logic can be used to clarify a number of philosophical problems themselves (...) ([HC72], p. x)

Among the problems raised by modal logics, the first is certainly the one concerning their own status as logics. Seen from the point of view of someone for whom there exists a thing such as The One And Only True Logic, which is the classical, two-valued one—modal logics are no more than mathematical formalisms, maybe nice to play with but without real philosophical importance. Witness for instance criticisms such as Quine's, for whom modal logics, first, were conceived in sin—the sin of confusing use and mention; and second, they are of no use anyway, because everything one does in some modal logic can be somehow translated into the formalism of first-order predicate calculus; and third, there are serious philosophical problems in their interpretation—among which one could mention a controversy over the interpretation of quantifiers (objectual vs. substitutional), as well as an apparent commitment of modal logics to essentialism (i.e., the thesis that objects have some of their properties essentially). But letting aside this dispute, which, however interesting, is out of this work's scope[4], modal logic's importance to the philosophical analysis of the notions of necessity and contingency should go without saying.

Considering now what is outside philosophy's realm, there is hardly any denying of the essential role played nowadays by logic itself in computer science and artificial intelligence (henceforth "AI")—but what about modal logics, particularly temporal and epistemic ones? Since we are going to talk a lot about epistemic logics in this work, I would like, before shifting attention to them, to say just some words about this other kind of modal logic, i.e., temporal (or tense) logics. There is again probably no need to stress their importance, at least not in philosophy:

> the theory of temporal logic is an integral concern of philosophical inquiry, and questions of the nature of time and of temporal concepts have preoccupied philosophers since the inauguration of the subject ([RU71], p. 1).

Among the contributions temporal logics can offer are formal models of time, which of course "provides the philosopher ... with tools for achieving a better understanding of the nature of time itself" ([RU71], p. 1).

In other areas, like computer science, the number of papers one can find dealing with, say, temporal logic of programs, is legion. It seems that computers, or at least logic programming, cannot dispose of a temporal logic of a kind—witness the following quotation (from the introduction of a paper of James Allen's, in which he presents an interval-based temporal logic):

[4] The reader wanting to know more about criticisms of modal logics can consult [Hck78], ch. 10.

> The problem of representing temporal knowledge and temporal reasoning arises in a wide range of disciplines, including computer science, philosophy, psychology, and linguistics. In computer science, it is a core problem of information systems, program verification, artificial intelligence, and other areas involving process modelling. ([Al83], p. 832)

As an example, if we are concerned with planning the activities of a robot, it is necessary to consider the effect of the robot's actions in the world, if they are to be effective. What involves the need to take changes into account, and changes obviously involve time. This is also emphasized in e.g. [MB83]: the authors state that "most work in AI which deals with *real world* problems would require some reasoning with time and space" (p. 343). Allen himself, in the mentioned paper, gives us more examples, such as databases which contain historical data—for example, if we are interested in modelling facts about the history of a person, we are bound to take time into account.

And so on. I'm not wanting to go into details at this point and on this subject, because, in spite of this work's title being very encompassing, not everything gets in. Temporal logics, for instance, are *not* mentioned—this work is about something else. The reader interested in this kind of modal logic can take a look at [Go87], where more examples are discussed, and whose emphasis is on computational matters, or at [Pr68] and [RU71], where more philosophical aspects are considered. So let us get down to the subarea of modal logics which is of special interest here (it *does* get mentioned on the title): epistemic logics.

0.2 Getting epistemic

First of all, it goes without saying that epistemic logic deals with epistemic notions, namely knowledge, belief, conviction, and other similar propositional attitudes. To put it In other words, epistemic logic is the kind of logic whose aim is "to explicate epistemic notions and to investigate the laws governing them" ([Len78], p. 16). Concretely, it is the kind of modal logic in which we interpret the box '□' as "A knows that ...", where A refers to some particular agent (which can be a human being, a robot, a knowledge base, a processor...). There is also a second side to this, namely the possibility of interpreting the box as "A believes/is convinced that ...", in which case we'd have a *doxastic* logic. The term "epistemic", however, usually covers both cases. And, instead of using the box, one commonly takes 'K' and 'B' to symbolize the desired operators. Sometimes these notational changes are the only ones we have: the axioms and inference rules of some alethic system are kept as paradigms. For instance, very often the modal calculus S5 is taken to be the logic of knowledge (e.g. in [HM84]), and weak S5 (a.k.a. KD45) to be the logic of belief.

This briefly sketched situation describes only the case in which we consider a single agent. But it is common in AI to have situations in which one must consider a whole lot of interacting agents. So, if one has, say, $1,\ldots,m$ agents, one has to introduce one operator K_i for each of these agents.

On the semantical side, when we now talk about possible worlds we are no longer having in mind some metaphysical sense of possibility, but rather what the agents think to be possible. The terminology "possible worlds" is even replaced by "epistemic alternatives", meaning the different ways the world can be according to the agent. Thus an agent knows some proposition p iff p is true in ali worlds she thinks are possible. In a sense, we still are talking about possibility, but now a subjective one. As a side remark, there is a lot of discussion about whether one can believe impossible things, that is, whether only *logically possible* worlds are allowed to count as sound epistemic alternatives, or whether we could, maybe, have some impossible ones, too. Opinions are greatly divided. (More about this question e.g. in [Len78] or [Hi75].)

Now what is, concretely, the importance of epistemic (or doxastic) logics, aside from a purely philosophical one? Well, their role is central to research in artificial intelligence, but not only there: examples in economics, linguistics, computer science, etc., are easy to find. Let us keep to the AI case. According to Stanley Rosenschein, *everything* in AI has to do with knowledge. For instance, he states that the major subareas of AI can be described in a way that highlights the importance of the concept of knowledge. I quote:

> - **Perception** has to do with an agent's acquiring *knowledge* about its environment by interpreting sensory input.
> - **Planning** has to do with an agent's acting on the basis of its *knowledge* of the consequences of its potential actions.
> - **Reasoning** involves an agent's deriving conclusions from facts it already *knows*.
> - **Learning** involves incrementing *knowledge* through experience.
> - **Communication** (e.g. in natural language) involves the continual updating of mutual *knowledge* possessed by the speaker and hearer. ([Ro85], p. 3)

So it seems that one cannot deny the importance of treating knowledge in AI—Rosenschein even speaks of the existence of a "knowledge industry" ([Ro85], p. 3). Now, since epistemic logic is a topic about which this work is concerned, perhaps we could talk somewhat more about its importance by considering some more concrete examples. In doing this, I'll closely follow a nice paper of J. Halpern's [Ha86b], in which the author addresses these questions. There he talks about the importance of reasoning about knowledge in certain areas of research in AI, like distributed systems, logical omniscience, common knowledge, knowledge and action. (To these I would also add nonmonotonic logics.) I'll try to characterize briefly the importance of epistemic matters in each one of these topics, what shall give us a little more of the flavor of this subject.

1. Distributed Systems

Distributed systems of computers, as one can grasp by taking a look at the specialized literature, are becoming more and more popular and widely applied. Such systems are used, for instance, to compute a protocol, which is "an algorithm whose execution is shared by a number of independent participants" ([LR86], p. 208). More precisely, a distributed system can be characterized as follows:

> A distributed system consist of a collection of processors, say 1,...,m, connected by a communication network. The processors communicate which each other over the links in the network. Each processor is a state machine, which at all times is in

> some state. This state is a function of the initial state, the messages it has received, and possibly some internal events (such as the ticking of a clock). ([Ha86b], p. 5)

In other words, we have the different participants in the system computing different tasks, and, contrary to sequential or parallel processing (where processors share the same memory), each player doesn't necessarily know what the others are doing, even if, in fact, they are exchanging messages all the time. (This is for instance a reason why the former logics of programs are inadequate when we reason about the behavior of protocols. Cf. [LR86], p. 208.) Now this property— that players are not necessarily aware of what the others are doing—characterizes just the *lack of knowledge* from each player with respect to the total state of the system. According to [Ha86b], the notion of knowledge at stake here is an "external" one, meaning it is not the *processor* who thinks ("and scratches its head") about whether or not it knows something, but it's rather a *programmer*, from an outside point of view, who says that the processor knows, or not, some fact. Even though, one cannot dispute that reasoning about knowledge is a very important characteristic of distributed systems. Quoting from [LR86], "any logic of protocols must include as part of it a logic of knowledge" (p. 208).[5]

Now talking about knowledge in situations involving more than one agent involves a lot of "subtleties" ([HM86], p. 1). The point can be better illustrated by the following *puzzle of the muddy children* ([HM86], p. 2):

> Imagine *n* children playing together. The mother of these children has told them that if they get dirty there will be severe consequences. So, of course, each child wants to keep clean, but each would love to see the others get dirty. Now it happens during their play that some of the children, say *k* of them, get mud on their foreheads. Each can see the mud on others but not on his own forehead. So, of course, no one says a thing. Along comes the father, who says, "At least one of you has mud on your head", thus expressing a fact known to each of them before he spoke (if $k > 1$). The father then asks the following question, over and over: "Can any of you prove you have mud on your head?" Assuming that all the children are perceptive, intelligent, truthful, and that they answer simultaneously, what will happen?
>
> There is a "proof" that the first $k - 1$ times he asks the question, they will all say "no" but then the k^{th} time the dirty children will answer "yes".

I'm not going to discuss the "proof" here; the reader is referred to [HM86], where the problem is examined in detail. Now one of the "subtleties" this puzzle is suppose to illustrate is the following: since what the father said was *already known* by the children, it would seem that his statement wasn't needed at all. But this is not the case, the proof won't go without it ([HM86], p. 2). Thus, before and after the father's statement, we have two different situations with regard to what the children know. The difference involves the topic we are going to mention next: after the father's statement, the children have *common knowledge*.

[5] By the way, they state that a logic of time is also necessary.

2. *Common Knowledge*

Another theme that very often appears is discussions of knowledge, in particular in cases (as the one before) where more agents are involved, is the notion of *common knowledge*. To put it short, we say that a certain group of agents has common knowledge of a certain fact *p* not only (as one could think) if every member of the group knows *p*, but also if everybody knows that everybody knows that *p*, and if everybody knows that everybody knows that everybody knows that *p*, and if everybody knows ... That there is a big difference between the two situations is one of the points in the muddy children puzzle above. Without the statement of the father, even if every children knows that at least one has mud on his/her forehead, they don't have common knowledge.

Now, is common knowledge interesting? Can we find applications of it?

Sure. It seems that the notion of common knowledge is essential to the notion of agreement—"'agreement' implies common knowledge of the agreement" ([Ha86b], p. 10). We can see this clearly in the next example of the *coordinate attack problem* ([HM86], p. 6):

> Two divisions of an army are camped on two hilltops overlooking a common valley. In the valley awaits the enemy. It is clear that if both divisions attack the enemy simultaneously, they will win the battle, whereas if only one division attacks it will be defeated. The divisions do not initially have plans for launching an attack on the enemy, and the commanding general of the first division wishes to coordinate a simultaneous attack (at some time the next day). Neither general will decide to attack unless he is sure that the other will attack with him. The generals can only communicate by means of a messenger. Normally, it takes the messenger one hour to get from one encampment to the other. However, it is possible that he will get lost in the dark or, worse yet, be captured by the enemy. Fortunately, on this particular night, everything goes smoothly. How long will take them to coordinate an attack?

[HM86] show that, despite the fact that in the said night everything goes smoothly, it is impossible for the two generals to reach an agreement and coordinate an attack (p. 6). It is not difficult to see why: the first general will not attack unless he or she knows that the message proposing a joint action was delivered, and unless he or she knows that the other general knows that his or her acknowledgement of the first message was delivered, and unless ... Well, unless there is common knowledge that an attack is going to happen.

As a side remark to this, the authors in [HM86] show that "not only is common knowledge not attainable where communication is not guaranteed, it is also not attainable in systems where communication is guaranteed, as long as there is some uncertainty in message delivery time" ([Ha86b], p. 10). This also holds for humans—think for instance of how often, and under which difficult conditions, do nations reach agreements...

3. *Knowledge and Action*

It is common, I think, that examples intending to illustrate some point end up throwing light in more than one. In the previous example of the coordinate attack, not only common knowledge is at stake, but it also involves communication and acting upon having knowledge. Knowledge and action, for instance, are crucially intertwined: "Knowledge is necessary to perform actions, and new knowledge is gained as a result of performing actions" ([Ha86b], p. 11). Witness also the quotation from [Ro85] above, concerning planning: an agent acts based on the knowledge of the consequences of its actions. Discussions of this topic can be found in [McH69], and also in a paper by R.C. Moore ([Mo81]), where he introduces a logic combining knowledge and action. The main point is that knowledge alone, or reasoning about knowledge alone, is of little value—mostly we are interested in having information about what we can do with the knowledge we've got. To mention the example discussed in [Mo81], if there is a safe that John wants to open, we might make the following inferences: if John knows the combination, he can immediately open the safe. Or, if he doesn't know the combination, but know where it is written, he can read the combination and then open the safe ([Mo81], p. 473). As one can see, if John doesn't have knowledge about the combination, his next preoccupations could well be how to obtain this knowledge, and which actions are necessary for that.[6]

4. Logical Omniscience

After having stressed (I hope successfully) the importance of the notion of knowledge in different topics in AI, one should also mention some problems concerning the way things are being done. I said over and over again that epistemic logics are important. Now, are the ones we have really good for the uses we have in mind for them? Yes and no. That is, there is a problem with the way people model knowledge, and the key words here are "logical omniscience".

If one would take a day out to dive in the literature concerning epistemic logic, one would surely notice that people have been and are still talking a real lot about this problem. One would also notice that most authors just *avoid* the problem ("we consider in this paper only agents having very powerful reasoning capabilities..."), or they accept logical omniscience as a kind of *malum necessarium*, that is, something unpleasant you have to live with if you want to have a logical system at all. Not so often (but it has been changing in the last years), people do try to find a solution to the problem.

To present things in an informal way, an agent A is considered to be logically omniscient if she knows all logical consequences of her knowledge, among which, by the way, are all "logical truths". In other words, if A knows that A, and moreover B is a logical consequence of A, then A knows that B, too. Or take A to be a logical truth, e.g. a tautology: then A also knows that A.

Now this presentation is surely very rough—there are many other, more rigorous formulations of the problem—but it will be enough for our motives here. (A more detailed presentation of several different "formal encodings" of consequential closure principles can be found in [Len78], pp. 53ff.)

More often than not, this situation concerning an agent's reasoning capabilities is considered to be a plague. I am not wanting to commit myself here and yet on this topic—one should, first of ali, better check out whether this situation is really a Bad Thing, or something that's not really that serious. But anyway, one should at least have the possibility of making choices, that is, it would be nice to have different epistemic-doxastic logical systems in which one can have or not, as one likes, logical omniscient agents. This is exactly the reason why logical omniscience is seen as a plague for the possible-world semantics:

[6] One couldn't have made this point more precisely than "Slippery Jim" diGriz, the Stainless Steel Rat: "Money was what I wanted. Other people's money. Money is locked away, so the more I knew about locks the more I would be able to get this money." (Harry Harrison, *A Stainless Steel Rat is Born*. Ney Work, Bantam Books, 1988, p. 12.)

you *don't* have a choice. Because of the way this semantics is built up, agents end up being logically omniscient—or so they seem. Remember above, where we said that **A** knows p iff p is true in every world (epistemic alternative) that **A** thinks to be possible? Well, worlds are supposed to be logically consistent—they are *logically possible* worlds—so tautologies are bound to be true in every conceivable one. Hence, modelling an agent's knowledge in this way is to assert, right from the beginning, that she's going to be logically omniscient. There's simply no world in which a tautology A will be false, so that we can falsify $\mathrm{K}_{\mathbf{A}}A$ too.

There are of course tentative solutions, even if not very satisfactory ones—sometimes one ends up with mixings of syntax and semantics; or one finds out that agents are no more logically omniscient with regard to the classical logic, but they are, say, in some relevance logic, what doesn't looks much better and also doesn't seem to agree with our intuitions. But see for instance [Hi75, Lev84, Va86] on the problem. And more about this on [Ha86b], which also mentions additional bibliography.

5. *Nonmonotonic Logics*

To close this general introduction and background painting, I have to say a few words about nonmonotonic logics (henceforth NMLs), not only because one of the approaches to the formalization of nonmonotonic reasoning makes use of epistemic logics, but also because our work touches marginally upon these matters, or better: this work's starting point arose from some problems in epistemic logics that deal with the formalization of nonmonotonic inferences. But what exactly are NMLs? Or, for that matter, what is monotonicity?

A characteristic of classical logic—to tell the truth, also of a lot of its rivals—is the following one: once you have carried out a valid inference, *nothing* you may possibly add to your premisses will ever change the validity of this inference (even if you add the negation of the conclusion, because in this case the premisses would be inconsistent and anything goes). Now this is what monotonicity is about: by adding premisses you can gain information (in the form of new conclusions), but you lose nothing. A little more formally, if p follows from some set Γ of propositions, monotonicity guarantees that it also follows from Γ augmented by whatever set of new propositions you like.

Well, if one now thinks about how things should ideally be, inferences being monotonic seems to be something desirable in logic. However, more often than not, we humans are confronted with situations in which we have to *refrain* from previously derived conclusions. Consider the following proposition ([Gi87], p. 2): birds fly. It is obviously not the case that *all* birds fly, but they normally (*typically*) do. Now if someone tells you that Tweety is a bird, and you know nothing else about Tweety, you'll gladly jump (or even fly) to the conclusion that Tweety flies. But suppose afterwards you learn that Tweety has its feet set in concrete, old Chicago style: then you are not anymore ready to assert or believe that Tweety flies. So the inference from "Birds fly" and "Tweety is a bird" to "Tweety flies" was a nonmonotonic one: upon learning new information, we have to retract this conclusion. Actually the inference relied more on the fact that *typical* birds fly, and, in the absence of contrary information, you assumed that Tweety was a typical bird, from what you refrained upon learning of its predicament.

Now there are several different ways of doing nonmonotonic reasoning, or, to put it better, of trying to formally capture such inferences—like default logics, or circumscription. I won't discuss all them possibilities here, but I have to mention the one among these which is of some interest to us here: the so-called "modal approaches" (cf., also what follows, [Gi87], pp. 8–9).

According to Ginsberg, the first ones who tried to use a modal logic to model nonmonotonic reasoning were McDermott and Doyle [McD80], in which they used a first-order logic augmented by a modal operator *M*, which should mean "maybe" or "is consistent with everything else that is known". If we take '*b*' to symbolize 'Birds fly', and '*f*' to 'Tweety flies', our example inference could be then formalized as

$$b \wedge Mf \rightarrow f.$$

This *M* operator, however, is not entirely without problems. According to R.C. Moore, it characterizes so weak a "notion of consistency that, as [McDermott and Doyle] point out, *MP* is not inconsistent with $\neg P$" ([Mo83], p. 128). Moore set out to change this himself, what he accomplished by introducing an *autoepistemic logic*, in which he changed this weak consistency operator into an epistemic necessity operator ("it is known" or "it is believed"). So the example inference would now be formulated as

$$b \wedge \neg L \neg f \rightarrow f.$$

With such an approach we reach then a state of things where also epistemic logics can make an important contribution to the formalizations of nonmonotonic reasoning. But this is all what I wanted to say about NMLs here. I hope I could have made clear the importance of epistemic notions in AI and computer science, and also that I succeeded in giving an idea of where our work is going to fit in. So now let us get down to our specific research problems here.

0.3 Down to specifics

The main interest of this work—its leading thread, or, at least, where things begin—comes from epistemic logic, more precisely, from the tentative of characterizing minimal belief states. The problem can be traced back to a paper of J. Halpern and Y. Moses' called "Towards a theory of knowledge and ignorance" ([HM84]). In said paper the authors (henceforth HM) consider the problem of characterizing the knowledge state of an agent A in situations where A has only partial information about some domain, that is, when A knows only some formula α. In their paper they assume that reasoners are logically omniscient, i.e., that they are perfect reasoners concerning propositional logic: they know all logical consequences of what they know. Besides, reasoners are also thought to have perfect introspective knowledge about their own knowledge or ignorance: they are completely in clear about what they know and what they don't know. As a consequence of these assumptions, the characterization of a knowledge state is a non-trivial matter. Suppose, for example, that A(ngela)[7] knows only *p*: she can discover by introspection that she doesn't know *q* and thus she knows she doesn't know it. This entails that something more than just the logical consequences of knowing α belongs to an agent's knowledge state, and so any attempt to

[7] It's a little bit boring and cumbersome to speak all the time about "an agent A", so I prefer to give her a name, like Angela. Things look nicer this way, too. For the reason why we're talking about female agents, cf. Lazarus Long: "Men are more sentimental than women. It blurs their thinking."

characterize such a state will have to take this fact into consideration. To cope with the problem, HM present different characterization methods, making use for instance of Kripke models, stable sets, and so on. They also introduce the notion of an "honest" formula, namely of a formula that uniquely characterizes Angela's knowledge state, when this formula is everything that she knows. As an example, or rather as a counter-example, the formula $\alpha = \mathrm{K}p \vee \mathrm{K}q$ is not honest, because an agent cannot know α without knowing either p or q. On the other hand, $\mathrm{K}p \wedge \mathrm{K}q$ is honest. HM present several ways of defining honesty, and they are all proved to be equivalent. An algorithm for deciding about the honesty of a formula is also given.

So far so good, but the logic used in [HM84] is propositional S5, and, as I mentioned, this implies that agents are supposed to be logically omniscient and fully introspective. What can be just fine for a lot of applications, but using S5 as a logic of knowledge will give us some problems the moment we try to formalize in it nonmonotonic inferences, for instance, default reasoning. Loosely speaking, a default rule could be described as follows: q (which is called the *default*) is true, unless one knows that p is true (cf. [HM84], p. 17). Formally, $\neg\mathrm{K_A}p \rightarrow q$, where '$\mathrm{K_A}p$' is to be understood as 'Angela knows that p'. Now according to HM the formula $\neg\mathrm{K_A}p \rightarrow q$ is itself not honest, if p and q are propositional variables; further, the authors state that such a formula doesn't behave at all like a default rule:

> In fact, for an honest α, $[\neg\mathrm{K_A}p \rightarrow q]$ is a consequence of 'knowing only α' exactly if one of p or q is ([HM84], p. 17).

As a consequence one should, in their opinion, either give up on the hope of having consistent non-monotonic default rules, or else give up on S5 as an adequate logic for modelling knowledge. HM, however, would like to preserve both, so they suggest, as a possible way of escaping this dilemma, that default rules could perhaps be better formalized in an *epistemic-doxastic logic*, namely as formulas of the form $\mathrm{B_A}(\neg\mathrm{K_A}p \rightarrow q)$, where '$\mathrm{B_A}$' stands for 'Angela believes that ...'. They justify this suggestion by saying that

> it is not our knowledge or ignorance of p that makes q true, but it is our information regarding our knowledge-gathering capabilities that leads us to believe q in the absence of our knowledge of p ([HM84], p. 17).

(Under "epistemic-doxastic logic"—EDL for short—we understand a logic system in which *both* concepts, knowledge and belief, are contemplated.)

So, to begin this work, and to set its main and more important goal, I will follow their suggestion and try to characterize "minimal epistemic states" in different EDL-calculi. In other words, I'll be looking for ways of describing Angela's epistemic (i.e., knowledge or belief—we'll decide it later) state under the supposition that she knows or believes only some formula α. In doing this I won't stay restricted to the only EDL-system proposed by HM (since this logic's "knowledge branch" is S5, and I am not that convinced that S5 is the best option in formalizing knowledge), but I'll rather try to work with several calculi of different strength.

Thus, in Chapter 1, we'll have an overview of some epistemic-doxastic logics. I will introduce several systems which we'll be working with, giving for each one an axiomatic presentation. This will be accompanied by a small discussion about the tenability of the various epistemic-doxastical principles involved. Also in the syntactical part are included some results about the number of modalities and the existence or not of reduction laws with respect to each system, results which will prove to be useful later on. In the semantical part we'll introduce a possible-world semantics for our logics, after what correctness and completeness theorems will be proved.

Chapter 2 will be devoted to the characterization of minimal epistemic states, in which I'll try the different approaches already employed by HM. A first trial employs stable sets, which we'll use to represent epistemic states. A next, short section will establish some relation between stable and saturated sets (which are maximal consistent), giving an alternative to our characterization problem. A third approach will rely on Kripke models—that is, on possible-world models; and last (but not least) an algorithmic approach. We'll see that each of these methods yield different results, depending on the logic being considered—sometimes they work, and sometimes don't. Nicely enough, the algorithmic one will prove to be the most general of them. Now, to talk a little bit more about the importance of this enterprise, I would like to mention that HM's original motivation arose from the question of "how communication in a distributed system changes the state of knowledge of the processors in the system" ([HM84], p. 1). I already remarked above, speaking of distributed systems, that the players in the system suffer under a lack of knowledge concerning what the remaining players are doing. This is an example of a situation where agents only have *partial* information at their disposal. In other words, everything they know (or believe) can be described by some formula α. The question of how to characterize an agent's knowledge state in such a situation comes from our intuition that there must be one, and only one, of such states, which fully describes what the agent knows (or believes). In our case here, where we consider both knowledge and belief, we have the additional motivation that results could also be of use for the formalization of nonmonotonic reasoning.

As I was saying, then, the algorithmic method is the one which will prove to be the most fruitful. In the case of HM's paper, the algorithm relies in a decision procedure for S5, which was the knowledge logic assumed there. Here we'll have obviously to examine decision procedures for each of the epistemic-doxastic logics we are considering. This new goal makes the connection to the second part of the present work, in which I consider valuation semantics and generalized truth tables for alethic modal logics. I hope I have already made a case concerning the importance of alethic modal logics in the preceding sections of this chapter. At the risk of repeating myself, the structure of modal logics and the EDLs considered here is very similar; thus we can adapt results in alethic modal logics to the epistemic case. And I'll be taking a look at valuation semantics because they easily yield decision procedures.

We'll thus begin Chapter 3 with a short and informal introduction to valuation semantics, trying to give the reader a first, intuitive idea of what they are about without jumping immediately to the definitions. In the following sections of this chapter we'll be examining the construction of such semantics for *normal* modal logics. These are, so to say, the mostly known among the modal logical systems, including landmarks such as T and the Lewis systems S4 and S5. We'll see that, for some of these logics, valuation semantics are (somewhat) easy to find, whilst for others we still are confronted with open problems.

In Chapter 4 will take care of *classical* modal logics, where here "classical" is not being employed in the sense I did some pages before, that is, meaning all modal logics which extend the classical one. In the sense of Chapter 4, classical modal logics are certain subsystems of the weakest normal one (which is K). These logics are, in a sense, of no greater importance to the main goal of this work, since the EDLs we'll be considering are ali normal, but it is nice to see in which way valuation semantics can be defined for other kinds of modal logic as well.

In Chapter 5 we will take a look at the main byproduct of valuation semantics, which are *generalized truth-tables*. These are, similar to the truth-tables for the classical propositional calculus, constructs which allow us to decide on the validity of a formula by examining the value it gets on different assignment of truth-values to its propositional variables and, depending on this, to its modalized subformulas. In other words, they are a method of having truth-tables—sort of—for modal logics. We'll show how by means of an example logic, K. And in the next chapter, which is number 6, we'll briefly look at how to obtain generalized truth-tables for one of the "problematic" normal logics of chapter 3, namely S4.

Having worked with generalized truth-tables, one has the feeling that there are some similitudes to another decision procedure for modal logics, the tableau systems. So in Chapter 7 we'll have a small comparison between the two methods, showlng that, to use a metaphor, they are the two sides of a coin—as are truth-tables and tableau systems for the classical propositional logic.

Finally, in Chapter 8, which closes the second part, we'll return to our main interest and show how to adapt valuations semantics, and hence the construction of generalized truth-tables, to the epistemic-doxastic logical case. We will take then an EDL as example, and adapt for it the whole procedure. As we'll see, this can be done in a more or less straightforward way: some semantic conditions are automatically given, because now all we have is for instance "Angela knows" instead of "necessarily". The conditions, however, that take care of the validity of "mixed" axioms—the ones involving knowledge and belief—are particular to EDLs, thus posing new problems. The reason is that in EDLs we have, so to say, *two* strong operators—i.e., which behave like necessity—and no weak operators. The opposite of your run-off-the-mill alethic modal logic.

The third part of this work concerns the "and their applications" part of the title, what reflects my interests in programming issues, in particular the implementation of theorem provers for modal logics. By "theorem proving" I mean of course *automated* theorem proving (henceforth ATP), which also goes by the name of "automated deduction". The big interest in ATP developed only in this century, with the Age of the Digital Computer, what gave researchers means of trying out their theoretical considerations. The ideas behind the ATP enterprise, however, are quite old, the automation of reasoning, or mechanizing of thought, being something many a philosopher or scientist dreamt about. Following M. Davis, we could say that the fundamental stone in the history of the mechanization of human thought was laid by Descartes with his employing of algebraic methods to develop classical Greek geometry: "what had seemed in Euclid to be the result of cunning and mathematical ingenuity was now revealed as being accessible to relatively mechanical treatment" ([Da83], p. 1). Descartes himself seemed to be quite aware of this, but the dream of doing for all deductive reasoning what he did for geometry was really born in the works of Leibniz, with his ambitious projects of a calculus of reason (*calculus ratiocinator*) and of a universal language (*characteristica universalis*). These projects, unfortunately, were never actually developed, for Leibniz had also many other interests: from the calculus of reason we have some fragments, but the universal language remained really a dream (cf. [Da83], p. 3). As I said, the real "boom" of interest in automated deduction really began in this century.[8]

Speaking of applications, there is of course, if one can say so, a more "theory oriented" side of this research: one would desire powerful ATPers in order to gain more knowledge in mathematics—either by obtaining new, maybe shorter proofs of known theorems, or even by proving propositions which now still

[8] A short but clearly arranged history of automated deduction—at least until the end of the 60's—can be found in Davis's paper [Da83].

have the status of conjectures. ATPers would be thus helping the progress of science. On a more "practical" side, again if one can say so, good ATP techniques can be used by computer scientists to prove properties of programs working on axiomatized structures (cf. [Ga86], p. 3). Not to forget applications in *logic programming*: let us consider the declarative language Prolog, for instance. A Prolog program consists in a set of facts and rules, that is, in a set of assertions, and a Prolog computation is in fact a proof, from which a program's output is to be extracted. It goes without saying that efficient proof techniques are vital to efficient implementations of Prolog. Speaking of this, there are extensions of the Prolog language which introduce modal or temporal features, in which case ATPers for modal logics also play an essential role when it comes to implementations.

Besides this, ATP techniques are of importance also in *database management*. It should be obvious that one cannot explicitly encode in a database all possible facts: a lot of them will have to be implicit. Take for instance the true proposition "the Earth has only one sun". It is also true that "the Earth doesn't have two suns", "the Earth doesn't have three suns", and so infinitely on. Since it is impossible to store ali these propositions explicitly, one has to make use of inference techniques to *deduce* such information. This is what also happens in the field of knowledge representation—e.g. concerning the knowledge base of some robot. One just cannot use memory to store every single fact such a robot knows or believes. Because—even if we set aside examples like the preceding—the robot is interacting with its ambient and "learning" new facts. In order to behave intelligently, it has to be able to draw inferences from pieces of information he gathers. So it makes sense to have some mechanism—and an efficient mechanism would be even better—which allows one to deduce new facts from facts already stored. This is where ATP techniques come into picture.

Thus, in the third part of this work, we'll try to go to the practical side of what we have so far discussed. In Chapter 9 we'll examine a program implementing generalized truth-tables for the example EDL of Chapter 8—a rather "naive" program, but reflecting with fidelity the definitions. The next Chapter, Chapter 10, will try to optimize this situation presenting a theorem prover which is an improved version of our first program, by using tableau proof techniques. As I remember mentioning before, generalized truth-tables and tableau systems can be seen as the two sides of a coin, so it is not surprising that, once we have GTT definitions for some logic, we can turn them around and generate tableau systems. I cannot go into much details here, because first we will have to see how, exactly, do our generalized truth-tables function. Finally, in Chapter 11, we'll implement, using the ATPer from Chapter 10, the algorithm used to characterize minimal belief states.

I

Minimal Belief States in Epistemic-Doxastic Logics

1

An Overview of EDL-Systems

I will now show off almost all the Greek I know:
"epistemic" has to do with knowledge; "doxastic", with belief.
So in what follows we shall have to do with
logics of knowledge and belief.

D. ISRAEL, *A Weak Logic of Knowledge and Belief.*

In this chapter I make a presentation of the epistemic-doxastic logics we are going to work with. We consider modalities and reduction laws, a possible-world semantics, and prove correctness and completeness.

1.1 Enter the logics

We'll use in this Part I a propositional language L which also includes operators for *knowing that* and *believing that*. Small letters ($p, q, r,...$) will be used as propositional variables, whereas capital letters, italicized, ($A, B, C,...$) will stand for syntactical variables for formulas. I'll also be occasionally using small greek letters as metavariables in some special cases, namely for formulas which denote everything that Angela knows or believes (like in "suppose α is everything that Angela believes"). Since we are going to keep to the one–agent case, 'KA' and 'BA' will be used as abbreviations for '$K_A A$' (that is, 'Angela knows that A') and '$B_A A$' (that is, 'Angela believes that A'), respectively. '$\neg$' and '$\rightarrow$' are introduced as primitive; the other boolean operators '$\wedge$', '$\vee$' and '$\leftrightarrow$' are defined in terms of these in the usual way. '**FOR**' stands for the set of all formulas of L.

We begin by considering a basic EDL[9] (at least in the bounds of this dissertation), which we will call Z. Actually, if we would follow the (more or less) standard way of christening modal logics (like in [Ch80]), this system should be named something like '$\mathbf{KT4K^bD^b4^b5^bM}$', where the '**KT4**' part refers to the knowledge branch, '$\mathbf{K^bD^b4^b5^b}$' to the belief one, and '**M**' to one "mixed" axiom (as one can see

[9] I'll use the expressions "EDL", "EDLogic", "EDL-calculus" and "EDL-system" as synonymous throughout this work.

from the axiom listing below). However, since we are here also going to consider several extensions of **Z**, names would be growing and growing, so let us agree (*non sunt prolonganda nomina præter necessitatem*, or words to that effect) that '**Z**' stands for **KT4K^{b}D^{b}4^{b}5^{b}M**. Thus, in this work, **Zxyz** will denote the extension of **Z** by adding schemas *x*, *y*, and *z* as axioms.

Now **Z** has the following axioms:

pc.	All tautologies of the classical propositional logic.	
k.	$\mathrm{K}(A \to B) \to (\mathrm{K}A \to \mathrm{K}B)$	
t.	$\mathrm{K}A \to A$	
4.	$\mathrm{K}A \to \mathrm{KK}A$	*(positive K-introspection)*
k^b.	$\mathrm{B}(A \to B) \to (\mathrm{B}A \to \mathrm{B}B)$	
d^b.	$\mathrm{B}A \to \neg\mathrm{B}\neg A$	
4^b.	$\mathrm{B}A \to \mathrm{BB}A$	*(positive B-introspection)*
5^b.	$\neg\mathrm{B}A \to \mathrm{B}\neg\mathrm{B}A$	*(negative B-introspection)*
m.	$\mathrm{K}A \to \mathrm{B}A$	

As rules of inference we have

MP. $\vdash A \,,\, \vdash A \to B \;/\; \vdash B$

RK. $\vdash A \;/\; \vdash \mathrm{K}A$.

As derived inference rules we can also have

RB. $\vdash A \;/\; \vdash \mathrm{B}A$

RKE. $\vdash A \leftrightarrow B \;/\; \vdash \mathrm{K}A \leftrightarrow \mathrm{K}B$

RBE. $\vdash A \leftrightarrow B \;/\; \vdash \mathrm{B}A \leftrightarrow \mathrm{B}B$

Let us now talk a bit about this axiomatization. The reader has surely noticed that the schema

5. $\neg\mathrm{K}A \to \mathrm{K}\neg\mathrm{K}A$ *(negative K-introspection)*

doesn't belong to the axiom basis, as it was perhaps to expect. There are two reasons for this, namely: (1) if we are *not* considering ideal agents (with regard to their introspective powers), *5* is clearly not valid (at least for human agents it is not, on what everybody seems to agree); and (2) if we put *5* together with some harmless-looking, acceptable EDL-principles, we get as a consequence lots of trouble (in the form of some nasty theorems, what I'll be showing soon enough). But *of course* we can take ideal agents in consideration, and thus extend **Z** by adding *5* as a new axiom. This resulting extension of **Z** we will call **Z5**.[10]

Z5 is actually the system HM suggested, but they mention it in a slightly different axiomatization—instead of *m* we would have the following axiom schema:

[10] I am not going here and yet to enter the discussion Ideal vs. Not Ideal agents, even if I'd like to do that. Hence we'll be suffering in this work of logical omniscience and similar troubles.

m^*. $KA \rightarrow BKA$.

We could call this other system Z5*, but it is not difficult to prove that both axiomatizations are equivalent, that is, they axiomatize the same logic. (See Appendix A1 for a proof of this claim.)

Now, lest the reader think that *5* is the villain of this story, it should be remarked loud and clear that not all the other axioms are accepted as evident. In fact, it seems that not a single one of them is free of criticisms for one or another reason. For the sake of completeness I am going to list some arguments against the different axioms.

Let us begin for instance with *k* and its belief instance k^b: these formulas can be said to embody one version of the infamous *principle of consequential closure*, which is just another way of spelling "logical omniscience" (or "logical omnibelief").[11] Letting these considerations apart, since we are taking for granted that Angela is omniscient, there are still other reasons—or what some researchers think to be reasons—why these principles shouldn't hold. k^b, for instance, should be plain false, if we interpret '**B**' not as conviction, but as a general, weaker kind of belief. In this interpretation, Angela believes some proposition *p* if she thinks *p* is more probable than its negation (cf. [Len78], p. 36). The argument against the validity of k^b is based on the *lottery paradox*, because we can show that k^b is equivalent to $BA \wedge BB \rightarrow B(A \wedge B)$, against which principle this paradox is directed. Suppose we have a lottery with, say, 1000 tickets, and let '$W(n)$' stand for "Lottery ticket number *n* is the wining one". Suppose also that Angela is buying a ticket. Since it is obvious that each ticket has only a very small probability of being the winning one, we can say Angela believes, for any *n*, that not–$W(n)$. More formally:

$$(*) \qquad B\neg W(1) \wedge B\neg W(2) \wedge \ldots \wedge B\neg W(1000).$$

On the other hand, it would be false to state that Angela believes the conjunction of these negated propositions:

$$(**) \qquad B(\neg W(1) \wedge \neg W(2) \wedge \ldots \wedge \neg W(1000)),$$

and this because she's *buying* a ticket; she is quite sure that *some* ticket *must* win (assuming it is a fair lottery, of course). We thus have a situation where Angela believes several propositions taken isolatedly, but not their conjunction. This only holds, of course, because we are here talking about weak belief—Angela is far from being convinced, of each ticket, that it won't win. Were this the case, she wouldn't obviously buy one. So the conviction analog of k^b holds, the same for *k*, the knowledge version.

However, it seems that one still could make a case for the validity of k^b, lottery paradox just the same. A possible way out of the predicament would be to say that even if we can believe, for *any n*, that ticket *n* won't win, this is not the same as believing it for *all n* (cf. [Har86], p. 71).

There are also some other tentatives of rejecting these principles, most running along the lines that someone knows some facts, and fails to draw its consequences—for instance with one of these logic puzzles that usually come in magazines. A reader can be said to know all the premisses, but mostly he or she needs a lot of time to arrive at the solution—if at all. This kind of example is actually not so good, because probably the agent doesn't know ("see") that $A \rightarrow B$, so it would be improper to assert $K(A \rightarrow B)$.

[11] The terminology "principie of consenquential closure" is due to K. Konolige (cf. [Ko86], p. 242). On logical omniscience, see the remakrs on Chapter 0 of this work. There are of course many other principies which entail logical omniscience, but they are somewhat beside the point here. A good discussion can be found in both [Len78] and [Len80].

Thus, everything considered, it seems hard to deny that some agent knows that *p*, and that *p* implies *q*, and nevertheless "fails to apply modus ponens" (cf. [Len78], p. 65).

Against *4* and 4^b there are also some proposed counterexamples. In the case of belief, they mostly refer to phenomena like ***subconscious*** beliefs and someone's not recognizing or even repressing such. (4^b was even acused of being "a short rejection of Freud". Cf. [Len78], p. 71.) Thus we have the ***atheist bishop*** example, which concerns a bishop who lost his faith. He believes that God doesn't exist, but cannot admit it to himself—thus he doesn't believe that he believes that God doesn't exist. The problem with this "counterexample" is, of course, the confusion between subcounscious beliefs and conscious ones. You can naturally choose which kind of belief 'B' is going to formalize, but then you've got to be consistent, what is not the case in the proposed couterexample.

Another proposed counter-argument states that, if I know that *p*, then I know that I know that I know that I know that I know that *p*, which is certainly somewhat unnatural. But the argument misses the point, because unnatural doesn't mean logically wrong. Besides, in certain systems, where we have reduction laws, such long iterations of epistemic operators can be proved to be equivalent to shorter, "natural" ones.

Similar attempts have been made to refute 5^b, mostly making the same mistakes (cf. [Len78], pp. 77ff). Its knowledge version, *5*, is, as I said, false, for the very simple reason that we commonly believe, or are even convinced that we know something, when it's not true. *5* would imply that we always can tell when we don't know something, and this is of course desirable, but highly unrealistic.

Against d^b it is said that it rules out incoherent or impossible beliefs, which many people (me for instance) seem to find desirable in certain contexts. Not in the sense that someone would or could believe a downright contradiction, like $A \wedge \neg A$, but maybe believe a set of facts, which, in the long run and in a non-obvious way, proves to be inconsistent.

The less disputed of the axioms is *t*, but even so there are some people who think it to be false, i.e., they defend one can know falsities (in which case I am prepared to concede that they *know* that *t* is false...) Mostly the arguments use the fact that we often "know" things that prove afterwards to be untrue, and generally there is some confusion between knowledge and knowledge claims.

And last a remark concerning the schema *m* ($\mathrm{K}A \rightarrow \mathrm{B}A$). Roughly speaking, this means that what Angela knows makes part of her beliefs (she believes what she knows). Putting things this way does lead to some confusion, and it is exactly on this confusion that some arguments against this principle are based. For instance, "I don't ***believe*** I'm married; I *know* it!" is the classical example. What is here at stake is, of course, a ***merely*** believing—I don't merely believe that I'm married, but of course I believe it. But one perhaps would like—and as a logician one should certainly try—to keep both concepts separated, in the sense that when one knows something, one doesn't actually believe it—one already knows it! We can introduce this concept without any problem in the calculi through the following definition:

$$\mathrm{B}^{*}A =_{\mathrm{df}} \mathrm{B}A \wedge \neg \mathrm{K}A.$$

But there are still other tentatives of rejecting this principle. Some of them (which in [Len78] are called "linguistic") concern the different uses of know and believe ([Len78], p. 24). For instance, it is entirely appropriate to say that I know Frankfurt, I know the name of the Bundeskanzler, and so on, but it doesn't make much sense to say that I believe Frankfurt, or Kohl's name. So *m* should be rejected. But *m* actually has to do with "knowing *that*..." and "believing *that*..." Finding out whether things like "I know Frankfurt" can be reduced to that-clauses is an open question.[12]

Another proposed argument against *m* considers an examinee who, being asked for certain historical dates, such as when did James I die, and being unsure about his knowledge, answers with "1635" while believing it is wrong date. It turns out, however, that "1635" is the correct answer. So the examinee *knew* the correct answer *without* believing it was correct. But of course the "argument" forgets that there is a difference between "he knew the correct answer" and "he knew that the answer was correct".

But let us leave the examples aside, at least for a while, and go back to axiomatizations. There are of course still other ways of extending **Z** besides **Z5**. We could for instance introduce one or more of the following "mixed" formulas as axioms:

$$p.\quad \mathrm{B}A \to \mathrm{KB}A$$

$$p^*.\quad \neg\mathrm{B}A \to \mathrm{K}\neg\mathrm{B}A$$

$$c.\quad \mathrm{B}A \to \mathrm{BK}A.$$[13]

Some words concerning them. First of all, *p* can be accepted without much problem: if Angela believes that *A*, then it is reasonable to suppose that she knows that she believes it. This can be justified by saying that Angela has a unique ("privileged") access to her own internal (epistemic) states. (But, as usual, there is a lot of discussion about this and related principles in the philosophical literature, mostly variations on the themes we've been discussing above.) *p** should be accepted on the same reasons. On the other hand, we can accept *c* only if we interpret 'B' not just as a kind of "weak belief"—like "I believe morning it's going to rain, but I'm really not that sure of it"—but as *conviction*. (It is a normal situation that there are a lot of propositions we believe, and nevertheless we are not willing to assert that we know them.)

Another possibility would be to make the knowledge branch stronger: not so much as S5, but, as many people like, as S4.2. We could do that by introducing the next formula as an axiom:

$$g.\quad \neg\mathrm{K}\neg\mathrm{K}A \to \mathrm{K}\neg\mathrm{K}\neg A.$$

If we now consider ali the possible extensions of **Z** by means of *p*, *p**, *c* and *g*, it seems on a first look that we would end up having something like 32 different logics. But this won't be the case, since, for instance, *p* and *p** are actually equivalent in **Z** (see Appendix A2); meaning it is enough to add one of these formulas as a new axiom in order to get the other one as a theorem.[14] We also get *g* as a theorem, if we add *5* or *p* as an axiom. So we get only the 9 following calculi:

Z:	*(the basic system)*
ZP:	$\mathbf{Z} + p = \mathbf{Z} + p^*$
ZC:	$\mathbf{Z} + c$
ZG:	$\mathbf{Z} + g$
ZCG:	$\mathbf{ZC} + g = \mathbf{ZG} + c$

[12] We are here, of course, moving in the realm of the so-called "Received View", where knowledge is actually (or reducible to) knowledge of facts: to say that I know an object is to say that I know facts about it. Cf. [Sa87], where this question is adressed and discussed in detail.

[13] I have been at great pains to find names for these schemas. So we'll have *m* because of "*m*ixed"; *p* because of "*p*rivileged knowledge of internal states"; *c* because of "*c*onviction", and *p** because it is equivalent to *p*.

[14] This doesn't necessarily hold if we are using a weaker belief logic (for instance without d^b).

ZCP: **ZC** + p = **ZC** + p^*
Z5: **Z** + *5*
ZP5: **Z5** + p = **Z5** + p^* = **ZP** + *5*
ZC5: **Z5** + c = **ZC** + *5* = **ZCP** + *5* = **ZP5** + c

The following diagram show us how these systems relate to each other. (An arrow means that the logic on the left is a subsystem of the logic on the right):

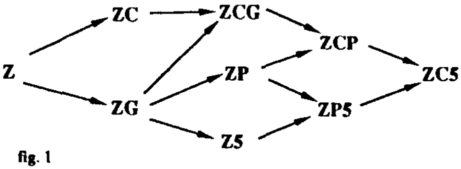

fig. 1

Looking closer at these systems, we see that adding *5* to **ZC**, or c to **Z5**, is enough to get **ZC5** (in which case g and both p and p^* are derivable, and this explains why this logic is just called **ZC5** instead of **ZCPG5**).

Let us now consider the problems I mentioned regarding axiom *5* (negative K–introspection): they appear in the systems **Z5**, **ZP5** and **ZC5**, particularly in **ZC5**. One can show that the formula B*KA* → *KA* is derivable in **Z5**—and this doesn't sound that reasonable: if Angela believes (or has the conviction) that she knows A, then she really knows it! This formula results from *5* together with d^b and m. Since we also have K*A* → BK*A* as a theorem, we can derive in **Z5** the equivalence BK*A* ↔ K*A*.

Well, this sure looks a good reason to forget schema *5*, or at least to have serious doubts concerning it—but, who knows, maybe for Angela to be convinced of knowing something is really the same as knowing that something.[15] The situation is still further complicated in **ZC5**. In this system we have B*A* → BK*A* as an axiom, and, from this schema, together with BK*A* → K*A*, we can prove B*A* → K*A*—and, what is (if possible) still worse, B*A*→*A* too! This of course means an equivalence between the knowledge and the belief operators, at the same time entailing that beliefs are infallible. At the risk of repeating myself, such a situation may be admissible if we are exclusively considering agents like Angela, for whom a notion of fallible bleief may make no sense at all. But if this were the case, we wouldn't like having these additional complications in the language of our systems—we would certainly prefer to make ourselves comfortably at home in a pure knowledge logic. In view of these considerations, I propose that we forget completely the unfortunate system **ZC5**, and work only with the other ones.[16]

[15] I'm not going to follow this question here, but maybe we can explain this strangeness. If Angela believes p, and doesn't know it, then, by *5*, she knows (and by m she believes) that she doesn't know that p. So it is not possible for Angela to believe that she knows p, against c.

[16] As a side remark on the psychology of ideal agents, we have here an interesting point: it seems that they cannot have *beliefs*, they *just know*. Because it should belong to the nature of belief, I think, that it can be defeated, that one isn't really sure that it holds. So ideal agents cannot believe. (Are they thus unable to have *faith*?)

We should also notice that there are still other ways of extending **Z**, **ZC**, **ZP** and **ZCP**, which I'm not going to consider: namely by using instead of *5* or *g* different characteristic axioms of the calculi between **S4** and **S5** (systems like S4.3, and so on).

To close this section, a small comparison between the logics I'm talking about in this paper and EDL-systems that were discussed by other logicians. The axioms and rules I have presented are well-known in the epistemic-logical literature, but a thorough study of its different combinations (like putting them together the particular way I'm doing here, with several logics of different deductive strength) seems, as far as I know, to be missing. (People commonly take one of the standard alethic systems and work with it.) **Z5**, of course, was already mentioned in [HM84].

In [Len80] we find the most complete study on epistemic-doxastic logics I know of, but Lenzen's presentation is somewhat different from my own here. First of all, he distinguishes, in the syntax of his logics, between *weak belief*[17] and *conviction*: to each of these concepts corresponds an operator (namely 'G' and 'Ü'), and there is of course an operator 'W' for knowledge. The principies we are taking here to hold of 'B' correspond to Lenzen's laws for the 'Ü'-operator.[18] So he has three operators, for the two in this work, in which I follow what he calls "the anglo-saxon tradition".

In the second place, Lenzen doesn't discuss systems of different strength but, in formalizing the logic of each concept, sets for the strongest possible calculus, i.e., a calculus that encompasses all the principles he considers to be valid (with regard to each concept). Thus Lenzen give us (mainly) 5 different systems, namely **G** (a pure logic of weak belief), **Ü** (a pure logic of conviction), **W** (a pure logic of knowledge), **D** (a combination of weak belief and conviction) and **E** (the strongest of his logics; the one containing the three operators). The logic **G** is somewhat weaker than **KD45** (which plays here the role of the belief logic); **Ü** is isomorphic to **KD45**, and **W** to **S4.2**.[19] Since I'm not making here a difference between weak belief and conviction, the logic **E**, if we leave 'G' out, corresponds to our system **ZCP**.

We are now ready to get things rolling. We can for instance define the notions of *proof* and *syntactical consequence* for our logics. We say that a sequence $A_1,\ldots,A_n$ of wffs is a *proof* in some logic L if, for $1 \leq i \leq n$, (i) A_i is an axiom of L; or (ii) there is $j < i$ and $k < i$ such that $A_k = A_j \to A_i$; or (iii) there is $j < i$ such that $A_i = \mathrm{K}A_j$. If $A = A_n$, we say that this sequence is a *proof of A in L* (and A is said to be a *theorem* of L, what we denote by $\vdash_L A$). If now Γ is a set of wffs, we say that A is a *syntactical consequence* of Γ in L (and we write $\Gamma \vdash_L A$) if there is a sequence $D_1,\ldots,D_n$ of wffs such that, for $1 \leq i \leq n$, (i) $A_i \in \Gamma$; or (ii) A_i is an axiom of L; or (iii) there is $j < i$ and $k < i$ such that $A_k = A_j \to A_i$; or (iv) there is $j < i$ such that $A_i = \mathrm{K}A_j$ and some subsequence of $D_1,\ldots,D_n$ is a proof of A_j. (Of course, $\nvdash_L A$ and $\Gamma \nvdash_L A$ mean that A is not a theorem of L, and is not deducible in L from Γ, respectively.)

Now, before we go into the next section, it is worth mentioning (later also worth using) that the following proposition holds of all our logics here:

Theorem T1. (*Deduction Theorem*) $\Gamma \cup \{A\} \vdash B$ *iff* $\Gamma \vdash A \to B$.

Proof In the usual deduction-theoremic way. ■

[17] Maybe "wider belief" would be a better name, since to Lenzen this notion ranges from a "mere surmise" (*bloße Vermutung*) to a "thorough coviction" (*feste Überzeugung*). (Cf. [Len80], p. 34)

[18] As I mentioned, Lenzen says that for instance *kb* doesn't hold if 'B' is taken to be a weak belief operator.

[19] Lenzen argues, by the way, that this calculus should be considered *the* logic of knowledge.

1.2 Modalities and reduction laws

In this section, as well as in the following one, I will be trying to obtain some results about the EDL–calculi which will be needed by the (tentative) characterization of epistemic states. These results concern primarily the number of modalities in each logic, and whether reduction laws are available. By a *modality* we understand any (finite) sequence of the operators $\neg$, K and B—this including the empty sequence $\bullet$, which is called the *improper modality*.

The first notion which is of importance here is the *modal degree* (dg) of a formula A, which we define as follows: if A is a propositional variable, then $dg(A) = 0$. If $A = \neg B$, $dg(A) = dg(B)$. If $A = (B \,\#\, C)$, for $\# \in \{$'$\wedge$', '$\vee$', '$\rightarrow$', '$\leftrightarrow$'$\}$, $dg(A) = max[dg(B), dg(C)]$. If $A = \mathrm{K}B$ or $A = \mathrm{B}B$, $dg(A) = dg(B) + 1$.

Next comes the defintion of a *modal conjunctive normal form* (MCNF): a wff A is in MCNF iff (a) the only operators that occur in A are '$\neg$', 'K', 'B', '$\wedge$' and '$\vee$'; (b) $A = D_1 \wedge \ldots \wedge D_n$ is in conjunctive normal form (like in classical propositional logic) and, for each disjunct D_i, either (i) $dg(D_i) = 0$, or (ii) $D_i = \#B$, where $dg(B) = 0$ and $\# \in \{\mathrm{K}, \mathrm{B}, \neg\mathrm{K}, \neg\mathrm{B}\}$.

We begin by examining each logic and trying to determine how many distinct modalities are there in it. In order to make things clearer to grasp, I am going to introduce here two abbreviations (in the same way as '$\Diamond$' abbreviates '$\neg\Box\neg$' in alethic modal logic):

$$\mathrm{P}A \quad =_{df} \quad \neg\mathrm{K}\neg A$$
$$\mathrm{C}A \quad =_{df} \quad \neg\mathrm{B}\neg A$$

Actually there seems to be no correct semantical interpretation for P and C,[20] but I think it is nice to use them as abbreviations, else one gets lost on a forest of negations. Of course our definition of a modality must be extended to contemplate this abbreviations too.

Let us now examine the different logics.

1. Modalities in Z

In **Z** we have a very little number of reduction laws. Since the knowledge branch is S4, we know that we have at most 14 pure knowledge modalities, and since the belief branch is **KD45**, we also know that we have at most 6 pure belief modalities. (Cf. [Ch80], p. 149, 154) But what happens with mixed sequences, like $\neg\mathrm{KB}\neg\mathrm{K}\neg\neg\mathrm{K}\neg\mathrm{BK}\neg\mathrm{B}\neg$ (or rather: $\neg$KBPPBKC)? Mixed reduction laws are not legion in **Z**. In spite of this, there is a finite number of distinct modalities in this logic, as we can see on the next theorem, even if it is very large:

Theorem T2. *In* **Z** *there are at most 84 distinct modalities, namely* $\bullet$, K, B, C, P, KB, KC, KP, BK/CK, PK, PB, PC, BP/CP, KBK/KCK, KPK, KPB, KPC, BKB/CKB, BKC/CKC, BKP/CKP, BPK/CPK, BPB/CPB, BPC/CPC, PKB, PKC, PKP, PBP/PCP, KPKB, KPKC, PKPB, PKPC, BKPK/CKPK,

[20] If we interpret 'B' as conviction, 'C' would mean (see [Len80], p. 16) something like "to think it possible that..." (*für möglich halten, daß...*). But there seems to be no correspondent in the case of weak belief, or knowledge, for that matter. But see [Hi62], where 'P' seems to mean something like "for all that one knows, ...". For an opinion against the existence of natural duals to 'K' and 'B', see [Isr], especially footnote 8 on p. 3.

BPKB/CPKB, BKPB/CKPB, BPKC/CPKC, BKPC/CKPC, BPKP/CPKP, BKPKB/CKPKB, BPKPB/CPKPB, BKPKC/CKPKC, BPKPC/CPKPC, *and of course the negations of them all:* ¬, ¬K, ¬B, *and so on.*

Proof. The proof is quite long, but relatively straightforward. It relies on the fact that the following equivalences (reduction laws) hold in **Z**:

(1) KKA ↔ KA
(2) PPA ↔ PA
(3) BBA ↔ BA
(4) CBA ↔ BA
(5) BCA ↔ CA
(6) CCA ↔ CA
(7) BKA ↔ CKA
(8) BPA ↔ CPA
(9) KBPA ↔ KPA
(10) KCPA ↔ KPA
(11) PBKA ↔ PKA
(12) PCKA ↔ PKA
(13) BKBKA ↔ BKA
(14) CKCKA ↔ CKA
(15) KBKBA ↔ KBA
(16) KCKCA ↔ KCA
(17) CPCPA ↔ CPA
(18) BPBPA ↔ BPA
(19) PCPCA ↔ PCA
(20) PBPBA ↔ PBA
(21) KCKBA ↔ KBA
(22) KBKCA ↔ KCA
(23) PBPCA ↔ PCA
(24) PCPBA ↔ PBA
(25) KPKPA ↔ KPA
(26) KBKPA ↔ KPA
(27) KCKPA ↔ KPA
(28) KPBPA ↔ KPA
(29) KPCPA ↔ KPA
(30) PKPKA ↔ PKA
(31) PKBKA ↔ PKA
(32) PKCKA ↔ PKA
(33) PBPKA ↔ PKA
(34) PCPKA ↔ PKA

Now we have obviously only one zero-lenght modality, which is •. We consider the other lengths, but just the positive cases:

(i) There are four modalities of length 1, K, B, C, and P, which we obviously cannot reduce further.

(ii) If we now add K, or B, or C, or P to the modalities of length 1, we get the following 16 modalities of length 2: KK, KB, KC, KP, BK, BB, BC, BP, CK, CB, CC, CP, PK, PB, PC and PP. Many of them, like KK, PP, BB, and so on, can be reduced using the equivalences (1) through (6). Moreover, with (7) and (8) we see that some other pairs are equivalent, even if not reducible. So we end with the following 8 modalities of length 2: KB, KC, KP, BK/CK, BP/CP, PK, PB, PC.

(iii) If we now repeat the procedure, we'll get, first, a lot of reducible modalities, like adding K to KB and obtaining KKB, which is immediately reducible to KB again. Trivial cases apart, we have equivalences (9) through (12), which also allow some further reductions. Thus we end with the following modalities of length 3: KBK/KCK, KPK, KPB, KPC, BKB/CKB, BKC/CKC, BKP/CKP, BPK/CPK, BPB/CPB, BPC/CPC, PKB, PKC, PKP, PBP/PCP, which we cannot further reduce.

(iv) There are 256 modalities of the lenght 4, but using the equivalences, now also the laws (13)-(34), we arrive at the following distinct 10 modalities: KPKB, KPKC, PKPB, PKPC, BKPK/CKPK, BPKB/CPKB, BKPB/CKPB, BPKC/CPKC, BKPC/CKPC, BPKP/CPKP.

(v) We repeat the procedure and obtain 4 modalities with length 5: BKPKB/CKPKB, BPKPB/CPKPB, BKPKC/CKPKC, BPKPC/CPKPC.

(vi) There are no irreducible modalities of the length 6. If we trying to expand the preceding modalities, trivial cases apart, we obtain things like KBKPKB, which, using (26), is equivalent to KPKB, which is of smaller length. And so with the other ones.

Now the negative cases are treated in a similar way, so the theorem is proven. ■

This list of all them modalities is everything we get here: I've been until now unable to draw a simple, easily understandable diagram depicting the relations between them. Some of the modalities, as you have noticed, come in pairs, like BK/CK: this means that they are equivalent, so I'm counting them as one. We can of course eliminate one of the pair, but we are unable to reduce them any further. On the other hand, one could take a different approach and say that modalities in a pair, even if equivalent, are (syntactically) distinct. In this case, Z would have 124 modalities.

2. *Modalities in* ZG

In **ZG** we have the additional axiom ¬K¬K*A* → K¬K¬*A* (or PK*A* → KP*A*), what allows us to reduce a little bit the number of modalities. As a result we end up with the following:

Theorem T3. *In* **ZG** *there are at most 46 distinct modalities, namely* •, K, B, C, P, KB, KC, KP, BK/CK, PK, PB, PC, BP/CP, KBK/KCK, KPB, KPC, BKB/CKB, BKC/CKC, BPB/CPB, BPC/CPC, PKB, PKC, PBP/PCP, *and of course their negations.*

Proof. The proof is similar to that of T2, with the difference that we now also have the following reduction laws:

(1) KPK*A* ↔ PK*A* (3) PKP*A* ↔ KP*A*

(2) BPK*A* ↔ PK*A* (4) CKP*A* ↔ KP*A*. ■

In the next picture we have an idea of the relations among **ZG** modalities. In the diagram, only the positive modalities are shown. To obtain the relation among the negative ones, just put a negation sign in front of each modality and then revert the direction of the arrows.

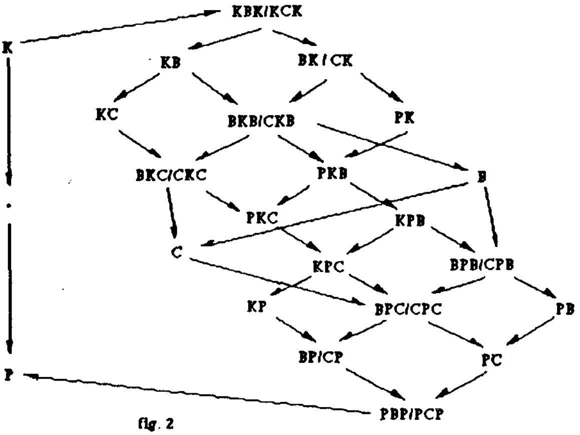

fig. 2

3. Modalities in ZC

In ZC things begin to get better. We have now as an axiom BA→BKA, and this allows us, since BKA→BA is already a theorem of Z, as well as BKA ↔ CKA, to substitute BK and CK everywhere for B. Actually we get the following new reduction laws:

(1) BKA ↔ BA	(3) PKA ↔ PBA
(2) CPA ↔ CA	(4) KCA ↔ KPA

which allow us to make big cuts in the number of Z modalities.

Theorem T4. *In* ZC *there are at most 18 distinct modalities, namely* •, K, B, C, P, KB, KC/KP, PB/PK, PC, *and their negations.*

Proof. As in T2. ■

Of these18 modalities, the 9 positive ones are in the following picture:

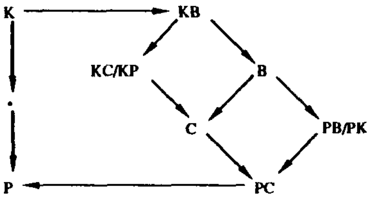

fig. 3

4. Modalities in ZCG

In ZCG we have *g* as an extra axiom; however, this doesn't allows us to reduce the number of modalities:

Theorem T5. *In* ZCG *there are at most 18 distinct modalities, namely* •, K, B, C, P, KB, KC/KP, PB/PK, PC, *and their negations.*

Proof. As in T2. ■

Now, even if the number of modalities is the same of ZC, the relations between them are other, what allows us to make the diagram simpler:

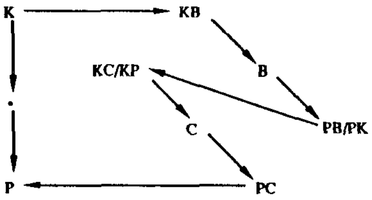

fig. 4

5. Modalities in ZCP

In **ZCP** things improve more. The situation is, by the way, very interesting here. Adding *p* as an axiom to **ZC**, or *c* to **ZP**, allows us now to derive $BA \leftrightarrow \neg K\neg KA$ as a theorem, what means that one could just do by introducing belief as a derived concept in a pure knowledge logic. Another point is that the knowledge branch is no more **S4**, since the axioms for belief (with 'B' replaced by '¬K¬K') now give us as theorem the **S4.2** characteristic axiom as well. So we are back to **S4.2**, and in this logic the results on number of modalities are well known (see e.g. [Ch80], p. 156 or [HC72], p. 261): there are ten of them, namely •, K, P, PK, KP and their negations. But since we want to keep belief in the picture, theorems like $BA \leftrightarrow PKA$ allow us to reduce the **S4.2** modalities even more (i.e., to only one epistemic operator each). We arrive at the end to the following result:

Theorem T6. *In* **ZCP** *there are at most 10 distinct modalities, namely* •, K, B, C, P, and their negations.

Proof. As in T2. ■

How they are related can be seen on the following picture:

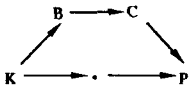

fig. 5

6. Modalities in ZP

In **ZP** we of course don't have everything as in **ZCP**, only part of it. Thus:

Theorem T7. *In* **ZP** *there are at most 14 distinct modalities, namely* •, K, B, P, C, BK/CK/PK, KP/BP/CP, *and their negations.*

Proof. As in T2. ■

The relations between the positive modalities are the following:

$$\begin{tikzcd}
K \arrow[r] \arrow[dd] & BK/CK/PK \arrow[d] \\
 & B \arrow[d] \\
\bullet \arrow[d] & C \arrow[d] \\
P & KP/BP/CP \arrow[l]
\end{tikzcd}$$

fig. 6

7. Modalities in Z5

Theorem T8. *In Z5 there are at most 18 distinct modalities, namely* •, K, B, P, C, KB, PB, KC, PC, *and their negations.*

Proof. As in T2. We now the reduction laws which hold in systems containing 5, namely PK*A* ↔ K*A* and KP*A* ↔ P*A*. ■

The relations among these (positive) modalities can be seen in the following diagram:

$$\begin{tikzcd}
K \arrow[rr] \arrow[dd] & & KB \arrow[dl] \arrow[dr] & & \\
 & KC \arrow[dr] & & B \arrow[dl] \arrow[dr] & \\
\bullet \arrow[d] & & C \arrow[dr] & & PB \arrow[dl] \\
P & & & PC \arrow[lll] &
\end{tikzcd}$$

fig. 7

8. Modalities in ZP5

ZP5, which is one of the strongest of our systems (the other being ZCP), will of course have very few distinct modalites.

Theorem T9. *In* ZP5 *there are at most 10 distinct modalities, namely* •, K, B, P, C, *and their negations.*

Proof. As in T2. ■

Now these are exactly the same modalities of ZCP, so all you have to do is to look at fig. 3 again!

Thus all of our systems have a finite number of distinct modalities. But as we know, this is not sufficient to guarantee that there also is only a finite number of different modal functions of one variable in each of the logics. In other words, this does not guarantee that we are able to reduce formulas, first, to the MCNF, and second, to a first degree one, what would be very nice. The modal logician reader certainly suspects that we won't find all we need in some systems, but quite probably in **ZP5**, which seems to be strong enough. We can in fact prove that in **ZP5** it is possible to reduce a formula from any degree whatsoever to another one of the first degree.

Proposition P1. *In* **ZP5** *we can reduce every formula of a degree higher than one to a first degree one.*

Proof. The proof of this proposition is somewhat long, but actually not difficult. (I'll sketch it here, details can be found in [Len80], p. 152ff, or in [HC72], p. 53ff.) The important point for the proof is the fact that in **ZP5** the following reduction laws are derivable, laws which allow us to eliminate iterated modalities:

(1) $\mathrm{KK}A \leftrightarrow \mathrm{K}A$

(2) $\mathrm{PK}A \leftrightarrow \mathrm{K}A$

(3) $\mathrm{BK}A \leftrightarrow \mathrm{K}A$

(4) $\mathrm{CK}A \leftrightarrow \mathrm{K}A$

(5) $\mathrm{KP}A \leftrightarrow \mathrm{P}A$

(6) $\mathrm{BP}A \leftrightarrow \mathrm{P}A$

(7) $\mathrm{CP}A \leftrightarrow \mathrm{P}A$

(8) $\mathrm{PP}A \leftrightarrow \mathrm{P}A$

(9) $\mathrm{KB}A \leftrightarrow \mathrm{B}A$

(10) $\mathrm{BB}A \leftrightarrow \mathrm{B}A$

(11) $\mathrm{CB}A \leftrightarrow \mathrm{B}A$

(12) $\mathrm{PB}A \leftrightarrow \mathrm{B}A$

(13) $\mathrm{KC}A \leftrightarrow \mathrm{C}A$

(14) $\mathrm{BC}A \leftrightarrow \mathrm{C}A$

(15) $\mathrm{CC}A \leftrightarrow \mathrm{C}A$

(16) $\mathrm{PC}A \leftrightarrow \mathrm{C}A$

(17) $\mathrm{K}(A \wedge B) \leftrightarrow \mathrm{K}A \wedge \mathrm{K}B$

(18) $\mathrm{B}(A \wedge B) \leftrightarrow \mathrm{B}A \wedge \mathrm{B}B$

(19) $\mathrm{C}(A \vee B) \leftrightarrow \mathrm{C}A \vee \mathrm{C}B$

(20) $\mathrm{P}(A \vee B) \leftrightarrow \mathrm{P}A \vee \mathrm{P}B$

(21) $\mathrm{K}(A \vee \mathrm{K}B) \leftrightarrow \mathrm{K}A \vee \mathrm{K}B$

(22) $\mathrm{K}(A \vee \mathrm{B}B) \leftrightarrow \mathrm{K}A \vee \mathrm{B}B$

(23) $\mathrm{K}(A \vee \mathrm{C}B) \leftrightarrow \mathrm{K}A \vee \mathrm{C}B$

(24) $\mathrm{K}(A \vee \mathrm{P}B) \leftrightarrow \mathrm{K}A \vee \mathrm{P}B$

(25) $\mathrm{B}(A \vee \mathrm{K}B) \leftrightarrow \mathrm{B}A \vee \mathrm{K}B$

(26) $\mathrm{B}(A \vee \mathrm{B}B) \leftrightarrow \mathrm{B}A \vee \mathrm{B}B$

(27) $\mathrm{B}(A \vee \mathrm{C}B) \leftrightarrow \mathrm{B}A \vee \mathrm{C}B$

(28) $\mathrm{B}(A \vee \mathrm{P}B) \leftrightarrow \mathrm{B}A \vee \mathrm{P}B$

(29) $\mathrm{C}(A \wedge \mathrm{K}B) \leftrightarrow \mathrm{C}A \wedge \mathrm{K}B$

(30) $\mathrm{C}(A \wedge \mathrm{B}B) \leftrightarrow \mathrm{C}A \wedge \mathrm{B}B$

(31) $\mathrm{C}(A \wedge \mathrm{C}B) \leftrightarrow \mathrm{C}A \wedge \mathrm{C}B$

(32) $\mathrm{C}(A \wedge \mathrm{P}B) \leftrightarrow \mathrm{C}A \wedge \mathrm{P}B$

(33) $\mathrm{P}(A \wedge \mathrm{K}B) \leftrightarrow \mathrm{P}A \wedge \mathrm{K}B$

(34) $\mathrm{P}(A \wedge \mathrm{B}B) \leftrightarrow \mathrm{P}A \wedge \mathrm{B}B$

(35) $\mathrm{P}(A \wedge \mathrm{C}B) \leftrightarrow \mathrm{P}A \wedge \mathrm{C}B$

(36) $\mathrm{P}(A \wedge \mathrm{P}B) \leftrightarrow \mathrm{P}A \wedge \mathrm{P}B$

It will be enough to show that we can reduce a second-degree formula to an equivalent first-degree one. The procedure that we use to accomplish this has four steps:

(1) We eliminate (by means of the definitions) the operators '→' and '↔'.

(2) Negation signs are pulled inside with the help of the DeMorgan and the reduction laws. At the end negations will occur just immediately before propositional variables.

(3) We reduce all iterated modalities, using the reduction laws, to a single modal operator.

(4) If the formula still is one of the second degree, the reason is that the formula itself, or one of its parts, is of the form #*B*, where # is a modal operator and *B* a conjunction or disjunction of the first degree. Using the laws (17) – (36) we can distribute and "absorbe" the # operator, so that at the end the result is a formula of the first degree. ■

Proposition P2. *In* **ZP5** *there is for every formula A an equivalent formula A^* such that A^* is a conjunction $D_1 \wedge \ldots \wedge D_n$, and each $D_i = \mathrm{K}B_1 \vee \ldots \vee \mathrm{K}B_m \vee \neg\mathrm{K}B_{m+1} \vee \ldots \vee \neg\mathrm{K}B_p \vee \mathrm{B}C_1 \vee \ldots \vee \mathrm{B}C_k \vee \neg\mathrm{B}C_{k+1} \vee \ldots \vee \neg\mathrm{B}C_j \vee E$, where $dg(B_1) = \ldots = dg(B_p) = dg(C_1) = \ldots = dg(C_j) = dg(E) = 0$ (i.e., A^* is in MCNF).*

Proof. First we eliminate from A implication and equivalence operators using the definitions. Then we examine the possible cases:

(a) If A is a zero-degree formula, we can simply reduce it according to PC-laws to the conjunctive normal form. It will be then automatically in MCNF (with the indices p and j being equal to zero in every case).

(b) Suppose now that A is a first-degree formula. Then it is a truth-function of wffs each of which is either a wff of PC or a wff of the form $\mathrm{K}B$, $\neg\mathrm{K}B$, $\mathrm{B}B$ or $\neg\mathrm{B}B$, where B is a zero-degree formula. Treating each of these formulas as if it were an atom, we reduce the whole formula to the conjunctive normal form by PC methods. The resulting formula is in MCNF.

(c) Suppose A is of a degree higher than one. Then we simply reduce it to a first-degree wff A', according to Proposition P1, and apply step (b) to this wff. ■

Now to **Z5**. We don't have in this system all the reduction laws from **ZP5**, but most of them. However, in **Z5** it is not possible to reduce every formula to a first-degree one. In spite of that, we get something which is almost as good for our purposes. We say that a wff A is a *β^0-formula* iff (i) $dg(A) = 0$; or (ii) for some B, such that $dg(B) = 0$, $A = \mathrm{B}B$ or $A = \neg\mathrm{B}B$. We can then prove the following proposition:

Proposition P3. *In* **Z5** *there is for every wff A an equivalent wff A^* such that A^* is a conjunction $D_1 \wedge \ldots \wedge D_n$, and each $D_i = \mathrm{K}A_1 \vee \ldots \vee \mathrm{K}A_m \vee \neg\mathrm{K}A_{m+1} \vee \ldots \vee \neg\mathrm{K}A_p \vee \mathrm{K}B_1 \vee \ldots \vee \mathrm{K}B_r \vee \neg\mathrm{K}B_{r+1} \vee \ldots \vee \neg\mathrm{K}B_q \vee \mathrm{B}C_1 \vee \ldots \vee \mathrm{B}C_k \vee \neg\mathrm{B}C_{k+1} \vee \ldots \vee \neg\mathrm{B}C_j \vee E$, where $dg(A_1) = \ldots = dg(A_p) = dg(C_1) = \ldots = dg(C_j) = dg(E) = 0$, and $B_1,\ldots,B_q$ are β^0-formulas.*

Proof. Similar to P2. ■

This of course amounts to saying that we can reduce a formula to one of the second degree.

The other six logics are complicated cases: we also don't have all of the **ZP5** reduction laws, just some, very few of them. We could now be hoping, since the number of, for instance, ZP-modalities is finite, that it would be possible, like we did in **Z5**, to reduce any **ZP**-formula to a second degree one. Actually, this is not the case. Remember, the "knowledge-branch" of **ZP** (all the formulas in which no 'B' occurs) is **S4**, and Makinson (see [Ma66]) has proved, for a supersystem of **S4** called **D**, that this system contains an infinite number of modal functions of one variable. Makinson's proof can be without much difficulty adapted for **ZP** and the other five logics, and so we come to the next proposition:

Proposition P4. *In* **Z**, **ZP**, **ZC**, **ZG**, **ZCG** *and* **ZCP** *there are infinitely many different modal functions of one variable.*

Proof. Using semantic methods; as it is in [Ma66], or in [Len80] pp. 241-43. ■

This result is also interesting with respect to ZCP: one could have hoped, because the modalities of **ZCP** and **ZP5** are the same, that the reduction laws would also be present. What is not the case, unfortunately.

1.3 A semantics for the EDL-systems

The goal of this third section is to provide each system with a possible world semantics, what we'll also be needing later. I will first define models for the basic system **Z**, and will show soon afterwards how to change the definition to obtain models for the other systems as well.

Definition D1. A Kripke model $\mathcal{M}$ for **Z** is a triple $<W,R,S>$, where:

a. $W \neq \emptyset$;

b. each $w_i \in W$ is an assignment of truth-values to each atomic formula;

c. $R \subset S \subset W \times W$;

d. S is reflexive and transitive;

e. R is transitive, serial and euclidean.[21]

The set W can be seen as a non-empty set of "worlds", or "points", or "epistemic states". To simplify things a bit I will consider them to be assignments of truth-values to propositional variables. R is the belief accessibility relation, and S the knowledge one.

We can now define, for each formula A, what it means for A to be true in a model and in a state:

Definition D2. Let $\mathcal{M} = <W,R,S>$ be a Kripke model, and w, v elements of W:

a. $\mathcal{M}, w \models A$ iff $w(A) = 1$, if A is a propositional variable;

b. $\mathcal{M}, w \models \neg A$ iff $\mathcal{M}, w \not\models A$;

c. $\mathcal{M}, w \models A \rightarrow B$ iff $\mathcal{M}, w \not\models A$ or $\mathcal{M}, w \models B$;

d. $\mathcal{M}, w \models \mathrm{K}A$ iff for every v, such that wSv, $\mathcal{M}, v \models A$;

e. $\mathcal{M}, w \models \mathrm{B}A$ iff for every v, such that wRv, $\mathcal{M}, v \models A$.

Now to obtain models for the other systems we need, as usual in possible-world semantics, to introduce some restrictions in the accessibility relations. To each new axiom there is a corresponding condition in the semantics that must be fulfilled:

g: S is incestual[22];

5: S is euclidean;

p: 1-mixed transitivity, that is: $\forall x \forall y \forall z(xSy \wedge yRz \rightarrow xRz)$;

[21] A relation R is *serial* iff $\forall x \exists y\, (xRy)$. R is *euclidean* iff $\forall x \forall y \forall z(xRy \wedge xRz \rightarrow yRz)$.

[22] A binary relation R is said to be *incestual* iff $\forall x \forall y \forall z(xRy \wedge xRz \rightarrow \exists w(yRw \wedge zRw))$.

p^*: mixed euclideanity, that is: $\forall x \forall y \forall z(xRy \wedge xSz \rightarrow zRy)$;[23]
c: 2-mixed transitivity, that is: $\forall x \forall y \forall z(xRy \wedge ySz \rightarrow xRz)$.

We now obtain models for the other logics just by restricting the accessibility relations S and R of the definition D1 in the following way:

ZG: S is also incestual;
ZP: 1-mixed transitivity (or mixed euclideanity);
ZC: 2-mixed transitivity;
ZCG: 2-mixed transitivity, S incestual;
Z5: S is also symmetric (or S is reflexive and euclidean);
ZCP: 1- and 2-mixed transitivity;
ZP5: S is symmetric, 1-mixed transitivity.

Based on this all we can now give the usual semantical definitions: a formula A is *true* in a model $\mathcal{M}$ for an EDL-calculus L ($\mathcal{M} \models_L A$) if, for every w in $\mathcal{M}$, $\mathcal{M},w \models_L A$. A is *L-valid* ($\models_L A$) if, for every L-model $\mathcal{M}$, $\mathcal{M} \models_L A$. A is in L a *semantical consequence* of a set Γ of formulas if, for every $\mathcal{M}$ and every $C \in \Gamma$ such that $\mathcal{M} \models_L C$, we have $\mathcal{M} \models_L A$.

It is now relatively easy to prove correctness and completeness theorems, as well as (but we won't do that here) the decidability of all systems. We begin by introducing the notion of a *saturated set* (what we'll also need later for the characterization of minimal states): a set Σ is an *A-saturated set*[24], for some formula A, if $\Sigma \nvdash A$ and, for ali $B \notin \Sigma$, $\Sigma \cup \{B\} \vdash A$.[25] Of course, a set Σ is *saturated* if, for some wff A, Σ is A-saturated.

Proposition P5. *Let Σ be a C-saturated set, for some wff C. Then:*

(a) $A \in \Sigma$ *iff* $\Sigma \vdash A$;
(b) $\neg A \in \Sigma$ *iff* $A \notin \Sigma$;
(c) $A \rightarrow B \in \Sigma$ *iff* $A \notin \Sigma$ *or* $B \in \Sigma$.

Proof. (a) In one direction, if $A \in \Sigma$ then obviously $\Sigma \vdash A$. In the other direction, suppose that $\Sigma \vdash A$. If now $A \notin \Sigma$, then by definition $\Sigma \cup \{A\} \vdash C$, but then it follows (by Cut) that $\Sigma \vdash C$, against the hypothesis that Σ is C-saturated. Hence $A \in \Sigma$.

(b) Suppose $\neg A \in \Sigma$. If we also have that $A \in \Sigma$, then Σ is inconsistent and is not C-saturated. So $A \notin \Sigma$. In the other direction, suppose that $A \notin \Sigma$. By definition, then, $\Sigma \cup \{A\} \vdash C$, and, by the deduction theorem, $\Sigma \vdash A \rightarrow C$. If now $\neg A \notin \Sigma$, we also have $\Sigma \cup \{\neg A\} \vdash C$, and, again by the deduction theorem, $\Sigma \vdash \neg A \rightarrow C$. But then $\Sigma \vdash A \vee \neg A \rightarrow C$; and, since obviously $\Sigma \vdash A \vee \neg A$, we have $\Sigma \vdash C$, against the hypothesis that Σ is C-saturated. Hence $\neg A \in \Sigma$.

[23] Since p and p^* are equivalent—at least in the systems considered here—it will be enough to add just one of the restrictions to the semantics.

[24] The notion of an A-saturated set was first used, as far as I know, by A. Loparić ([Lo77]).

[25] When there is no risk of confusion, I'm going to use '$\vdash$' and '$\models$' without subscripts.

(c) Suppose $A \to B \in \Sigma$. If $A \notin \Sigma$, there is nothing to prove. So let us suppose that $A \in \Sigma$. By (a), $\Sigma \vdash A$, $\Sigma \vdash A \to B$; so obviously $\Sigma \vdash B$ and, again by (a), $B \in \Sigma$. In the other direction, suppose first that $A \notin \Sigma$. So, by (b), $\neg A \in \Sigma$, and, since $\Sigma \vdash \neg A \to (A \to B)$, we have $\Sigma \vdash A \to B$, and $A \to B \in \Sigma$. Suppose then that $B \in \Sigma$. Since $\Sigma \vdash B \to (A \to B)$, we have $\Sigma \vdash A \to B$, and $A \to B \in \Sigma$. ■

Proposition P6. *If $\Gamma \nvdash A$, then there is an A-saturated set Σ, such that $\Gamma \subseteq \Sigma$.*

Proof. By a standard Lindenbaum argument. ■

To make life easier, let us introduce some abbreviations to talk about saturated sets. Let Γ be a set of formulas. We then define the subsets of Γ consisting of K- and B-formulas as follows:

$$\Gamma^{K} =_{df} \{A \in \Gamma\text{: there is } B, A = \mathrm{K}B\};$$
$$\Gamma^{B} =_{df} \{A \in \Gamma\text{: there is } B, A = \mathrm{B}B\}.$$

Next we define, for each of these sets, its *scope set*:

$$\varepsilon(\Gamma^{K}) =_{df} \{A\text{: } \mathrm{K}A \in \Gamma\};$$
$$\varepsilon(\Gamma^{B}) =_{df} \{A\text{: } \mathrm{B}A \in \Gamma\}.$$

Lemma L1. *If $\Gamma \vdash A$ then $\#\Gamma \vdash \#A$, where $\# \in \{\mathrm{K}, \mathrm{B}\}$ and $\#\Gamma = \{\#B : B \in \Gamma\}$.*

Proof. By induction on theorems. Suppose $\Gamma \vdash A$. We have four cases:

(1) $A \in \Gamma$. Then $\#A \in \#\Gamma$ and, obviously, $\#\Gamma \vdash \#A$.

(2) A is an axiom. Then $\vdash A$ and, by *RK* or (derived rule) *RB*, $\vdash \#A$, so $\#\Gamma \vdash \#A$.

(3) A was obtained by *MP* from B and $B \to A$. By the induction hypothesis, $\#\Gamma \vdash \#B$ and $\#\Gamma \vdash \#(B \to A)$. Since $\vdash \#(B \to A) \to (\#B \to \#A)$ (k or k^{b}), $\#\Gamma \vdash \#B \to \#A$, and hence $\#\Gamma \vdash \#A$.

(4) $A = \#B$ and was obtained by $R\#$. If $\Gamma \vdash A$, then there is a proof of A. By $R\#$, $\vdash \#A$, so $\#\Gamma \vdash \#A$. ■

Proposition P7. *If $\Gamma \nvdash \#A$, $\# \in \{\mathrm{K}, \mathrm{B}\}$, there is an A-saturated set Σ such that $\varepsilon(\Gamma^{\#}) \subseteq \Sigma$.*

Proof. If $\Gamma \nvdash \#A$, then obviously $\Gamma^{\#} \nvdash \#A$. By L1, $\varepsilon(\Gamma^{\#}) \nvdash A$ and, by P6, there is an A-saturated set Σ such that $\varepsilon(\Gamma^{\#}) \subseteq \Sigma$. ■

Before we go to the next bunch of properties that saturated sets have, let us define two binary relations ρ and μ over them. So let Γ and Σ be saturated sets; we say that

$$\Gamma \rho \Sigma \quad \text{iff} \quad \varepsilon(\Gamma^{B}) \subseteq \Sigma;$$
$$\Gamma \mu \Sigma \quad \text{iff} \quad \varepsilon(\Gamma^{K}) \subseteq \Sigma.$$

Proposition P8. *Let Γ, Σ and Δ be saturated sets. We have:*

(a) in all logics:

i. $\Gamma\mu\Gamma$;

ii. $\Gamma\mu\Sigma \wedge \Sigma\mu\Delta \Rightarrow \Gamma\mu\Delta$;

iii. $\exists\Theta\text{: } \Gamma\rho\Theta$;

iv. $\Gamma\rho\Sigma \wedge \Gamma\rho\Delta \Rightarrow \Sigma\rho\Delta$;

v. $\Gamma\rho\Sigma \wedge \Sigma\rho\Delta \Rightarrow \Gamma\rho\Delta$;

vi. $\Gamma\rho\Sigma \Rightarrow \Gamma\mu\Sigma$;

(b) in logics which have g as a theorem

vii. $\Gamma\mu\Sigma \wedge \Gamma\mu\Delta \Rightarrow \exists\Theta\text{: } \Sigma\mu\Theta \wedge \Delta\mu\Theta$;

(c) in logics which have p as a theorem:

viii. $\Gamma\mu\Sigma \wedge \Sigma\rho\Delta \Rightarrow \Gamma\rho\Delta$;

ix. $\Gamma\mu\Sigma \wedge \Gamma\rho\Delta \Rightarrow \Sigma\rho\Delta$;

(d) in logics which have c as a theorem:

x. $\Gamma\rho\Sigma \wedge \Sigma\mu\Delta \Rightarrow \Gamma\rho\Delta$;

(e) in logics which have 5 as a theorem:

xi. $\Gamma\mu\Sigma \wedge \Gamma\mu\Delta \Rightarrow \Sigma\mu\Delta$.

Proof. (i) We have to show that $\varepsilon(\Gamma^{K}) \subseteq \Gamma$. Let $A \in \varepsilon(\Gamma^{K})$; so $KA \in \Gamma$. Since $\vdash KA \rightarrow A$, $\Gamma \vdash A$, and $A \in \Gamma$.

(ii) Suppose $\Gamma\mu\Sigma$ and $\Sigma\mu\Delta$, and let $A \in \varepsilon(\Gamma^{K})$; thus $KA \in \Gamma$ and, since $\vdash KA \rightarrow KKA$, $KKA \in \Gamma$, and we get $KA \in \Sigma$. Since $\Sigma\mu\Delta$, $A \in \Delta$; thus $\varepsilon(\Gamma^{K}) \subseteq \Delta$ and $\Gamma\mu\Delta$.

(iii) Suppose there is no Θ such that $\Gamma\rho\Theta$; thus, if Θ is a saturated set, there is some wff A such that $BA \in \Gamma$ and $A \notin \Theta$. But then, since $\vdash BA \rightarrow \neg B\neg A$, $\neg B\neg A \in \Gamma$; $B\neg A \notin \Gamma$ and $\Gamma \nvdash B\neg A$. By P7, there is a $\neg A$-saturated set Θ such that $\varepsilon(\Gamma^{B}) \subseteq \Theta$. It follows that $\Gamma\rho\Theta$.

(iv) Suppose $\Gamma\rho\Sigma$ and $\Gamma\rho\Delta$, and suppose it is not the case that $\Sigma\rho\Delta$. Then there is some $A \in \varepsilon(\Sigma^{B})$ such that $A \notin \Delta$. Since $\Gamma\rho\Delta$, it cannot be that $BA \in \Gamma$, else A would be in Δ because $\varepsilon(\Gamma^{B}) \subseteq \Delta$. So $\neg BA \in \Gamma$ and, since $\vdash \neg BA \rightarrow B\neg BA$, $B\neg BA \in \Gamma$, and we get $\neg BA \in \Sigma$; but then Σ would be inconsistent. Hence $\Sigma\rho\Delta$.

(v) Like in (ii), using $BA \rightarrow BBA$.

(vi) Suppose $\Gamma\rho\Sigma$, and let $A \in \varepsilon(\Gamma^{K})$; thus $KA \in \Gamma$ and, since $\vdash KA \rightarrow BA$, $BA \in \Gamma$, and we get $A \in \Sigma$. Thus $\varepsilon(\Gamma^{K}) \subseteq \Sigma$ and $\Gamma\mu\Sigma$.

(vii) Suppose $\Gamma\mu\Sigma$, $\Gamma\mu\Delta$, and there is no Θ such that $\Sigma\mu\Theta$ and $\Delta\mu\Theta$. That is, there is no saturated set Θ such that $\varepsilon(\Sigma^{K}) \cup \varepsilon(\Delta^{K}) \subseteq \Theta$. By P6, then, for every formula A, $\varepsilon(\Sigma^{K}) \cup \varepsilon(\Delta^{K}) \vdash A$; i.e., $\varepsilon(\Sigma^{K}) \cup \varepsilon(\Delta^{K})$ is inconsistent. So there is a B such that, say, $B \in \varepsilon(\Sigma^{K})$ and $\neg B \in \varepsilon(\Delta^{K})$. Thus we have: $KB \in \Sigma$ and $K\neg B \in \Delta$; $\neg KB \notin \Sigma$ and $\neg K\neg B \notin \Delta$; $K\neg KB \notin \Gamma$ and $K\neg K\neg B \notin \Gamma$; thus $\neg K\neg KB \in \Gamma$. But $\vdash \neg K\neg KB \rightarrow K\neg K\neg B$, so $K\neg K\neg B \in \Gamma$, a contradiction. Hence there is a saturated set Θ such that $\varepsilon(\Sigma^{K}) \cup \varepsilon(\Delta^{K}) \subseteq \Theta$.

(viii) Suppose $\Gamma\mu\Sigma$ and $\Sigma\rho\Delta$, and let $A \in \varepsilon(\Gamma^{B})$; thus $BA \in \Gamma$ and, since $\vdash BA \rightarrow KBA$, $KBA \in \Gamma$, and we get $BA \in \Sigma$. Since $\Sigma\rho\Delta$, $A \in \Delta$; thus $\varepsilon(\Gamma^{B}) \subseteq \Delta$ and $\Gamma\mu\Delta$.

(ix) Like in (iv), using $\vdash \neg BA \rightarrow K\neg BA$.

(x) Like in (ii), using $\mathrm{B}A \rightarrow \mathrm{BK}A$.

(xi) Like in (iv), using $\vdash\neg\mathrm{K}A \rightarrow \mathrm{K}\neg\mathrm{K}A$. ■

Well, it certainly jumps to the eyes that these properties we've just proven saturated sets have are exactly the ones we require of the accessibility relations in the Kripke models for the different logics. We'll use all this later in the completeness proof.

Theorem T10. *(Correctness) If* $\Gamma\vdash A$ *then* $\Gamma\models A$.

Proof. Let us suppose that $\Gamma\vdash A$.

(A) $A \in \Gamma$. Then, for all $\mathcal{M}$ such that $\mathcal{M}\models\Gamma$, $\mathcal{M}\models A$.

(B) A is an axiom. We examine each case:

(*pc*) That is, A is tautology. Trivial.

(*k*) A is of the form $\mathrm{K}(p\rightarrow q)\rightarrow(\mathrm{K}p\rightarrow\mathrm{K}q)$. Let us suppose that A is not valid. Then there is a model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \mathrm{K}p$, $\mathcal{M}, w \models \mathrm{K}(p\rightarrow q)$ and $M,w \not\models \mathrm{K}q$. It follows that there is $v \in W$ such that wSv and $M,v \not\models q$. But it also follows that $\mathcal{M},v \models p$ and $\mathcal{M},v \models p\rightarrow q$, which is impossible.

(*t*) A is of the form $\mathrm{K}p\rightarrow p$. If A Is not valid, then there is a model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \mathrm{K}p$ and $\mathcal{M},w \not\models p$. However, since S is reflexive, wSw, and then it is not possible that $\mathcal{M},w \not\models p$.

(*4*) A is of the form $\mathrm{K}p\rightarrow\mathrm{KK}p$. If A is not valid then there is a model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \mathrm{K}p$ and $\mathcal{M},w \not\models \mathrm{KK}p$. Now it follows from D2.d that there is $v \in W$ such that wSv and $\mathcal{M},v \not\models \mathrm{K}p$. Again from D2.d we have a $t \in W$, such that vSt and $\mathcal{M},t \not\models p$. However, since S is transitive, wSt, so it cannot be that $\mathcal{M},t \not\models p$.

(k^b) A is of the form $\mathrm{B}(p\rightarrow q)\rightarrow(\mathrm{B}p\rightarrow\mathrm{B}q)$. Proof like in (*k*).

(4^b) A is of the form $\mathrm{B}p\rightarrow\mathrm{BB}p$. Proof like in (*4*).

(5^b) A is of the form $\neg\mathrm{B}p\rightarrow\mathrm{B}\neg\mathrm{B}p$. If A is not valid, then there is a model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \neg\mathrm{B}p$ and $\mathcal{M},w \not\models \mathrm{B}\neg\mathrm{B}p$; thus $\mathcal{M},w \not\models \mathrm{B}p$. From D2.e it follows that there is a $v \in W$ such that wRv and $\mathcal{M},v \not\models p$. From D2.e again we have a $t \in W$ such that wRt and $\mathcal{M},t \not\models \neg\mathrm{B}p$, hence $\mathcal{M},w \models \mathrm{B}p$. R is however euclidean, so we have that tRv, and then $\mathcal{M},v \models p$—a contradiction.

(d^b) A is of the form $\mathrm{B}p\rightarrow\neg\mathrm{B}\neg p$. If A is not valid, then there is a model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \mathrm{B}p$ and $\mathcal{M},w \not\models \neg\mathrm{B}\neg p$; thus $\mathcal{M},w \models \mathrm{B}\neg p$. Since R is a serial relation, there is a $v \in W$, such that wRv and $\mathcal{M},v \models p$. However, it follows from D2.b that $\mathcal{M},v \models \neg p$, $\mathcal{M},v \not\models p$, what cannot be.

(*m*) A is of the form $\mathrm{K}p\rightarrow\mathrm{B}p$. If A is not valid, then there is a model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \mathrm{K}p$ and $\mathcal{M},w \not\models \mathrm{B}p$. From D2.e there is then a $v \in W$, such that wRv and $\mathcal{M},v \not\models p$. However, since $R \subseteq S$, it follows from D2.d that wSv and thus $\mathcal{M},v \models p$—a contradiction.

(*g*) A is of the form $\neg\mathrm{K}\neg\mathrm{K}p\rightarrow\mathrm{K}\neg\mathrm{K}\neg p$. If A is not valid, then there is a ZG-model $\mathcal{M} = <W,R,S>$ and $w \in W$, such that $\mathcal{M},w \models \neg\mathrm{K}\neg\mathrm{K}p$ and $\mathcal{M},w \not\models \mathrm{K}\neg\mathrm{K}\neg p$; thus $\mathcal{M},w \not\models \mathrm{K}\neg\mathrm{K}p$. From D2.d it follows that there is a $v \in W$ such that wSv and $\mathcal{M},v \nvDash \neg\mathrm{K}p$, hence $\mathcal{M},v \vDash \mathrm{K}p$. From D2.d again we have a $t \in W$ such that wSt and $\mathcal{M},t \nvDash \neg\mathrm{K}\neg p$, hence $\mathcal{M},w \vDash \mathrm{K}\neg p$. S is however incestual, so we have a $u \in$ W such that vSu and tSu. It follows that $\mathcal{M},u \vDash \neg p$ and $\mathcal{M},u \vDash p$—a contradiction.[26]

(*5*) A is of the form $\neg\mathrm{K}p \rightarrow \mathrm{K}\neg\mathrm{K}p$. Proof like in ($5^b$), using now the fact that S is euclidean for **Z5** and **ZP5** models.

(*p*) A is of the form $\mathrm{B}p \rightarrow \mathrm{KB}p$. If A is not valid, then there is a **ZP**-model $\mathcal{M} = \langle W,R,S \rangle$ and $w \in W$, such that $\mathcal{M},w \vDash \mathrm{B}p$ and $\mathcal{M},w \nvDash \mathrm{KB}p$. From D2.d it follows that there is a $v \in W$ such that wSv and $\mathcal{M},v \nvDash \mathrm{B}p$. From D2.e it follows now that there is a $t \in W$ such that vRt and $\mathcal{M},t \nvDash p$. Now in **ZP**-models the 1-mixed transitivity holds, so we get that wRt, and consequently, that $\mathcal{M},t \vDash p$—a contradiction.[27]

(*c*) A is of the form $\mathrm{B}p \rightarrow \mathrm{BK}p$. If A is not valid, then there is a **ZC**-model $\mathcal{M} = \langle W,R,S \rangle$ and $w \in W$, such that $\mathcal{M},w \vDash \mathrm{B}p$ and $\mathcal{M},w \nvDash \mathrm{BK}p$. From D2.e it follows that there is a $v \in W$ such that wRv and $\mathcal{M},v \nvDash \mathrm{K}p$. From D2.d it follows now that there is a $t \in W$ such that vSt and $\mathcal{M},t \nvDash p$. Now in **ZC**-models the 2-mixed transitivity holds, so we get wRt, and consequently, that $\mathcal{M}$,t $\vDash p$—a contradiction.[28]

Thus, in all cases and all logics L, A is L-valid, and so, for all $\mathcal{M}$, such that $\mathcal{M} \vDash \Gamma$, $\mathcal{M} \vDash A$.

(C) A was obtained by using *MP* from B and $B \rightarrow A$. Induction hypothesis: for all $\mathcal{M}$, such that $\mathcal{M} \vDash \Gamma$, $\mathcal{M} \vDash B$ and $\mathcal{M} \vDash B \rightarrow A$. If $\mathcal{M} \nvDash A$, there is a $w \in W$ such that $\mathcal{M},w \nvDash A$. But $\mathcal{M},w \vDash B$, $\mathcal{M},w \vDash B \rightarrow A$, and this is contradictory. Thus, for all $\mathcal{M}$ such that $\mathcal{M} \vDash \Gamma$, $\mathcal{M} \vDash A$.

(D) $A = \mathrm{K}B$ was obtained from B using *RK*. Induction hypothesis: for all $\mathcal{M}$, such that $\mathcal{M} \vDash \Gamma$, $\mathcal{M} \vDash B$. Now if $\mathcal{M} \nvDash A$, there is a $w \in W$ such that $\mathcal{M},w \nvDash A$, i.e., $\mathcal{M},w \nvDash \mathrm{K}B$. From D2.d it follows then that there is a $t \in W$ such that wSt and $\mathcal{M},t \nvDash B$—and this cannot obviously be the case. Thus, for all $\mathcal{M}$ such that $\mathcal{M} \vDash \Gamma$, $\mathcal{M} \vDash A$. ■

To prove now the completeness theorem we need first to establish some relations between saturated sets and Kripke models. If $\mathcal{M}$ and $\mathcal{N}$ are Kripke models, we say that $\mathcal{M} \approx \mathcal{N}$ ($\mathcal{M}$ and $\mathcal{N}$ are *equivalent*) if, for every A, $\mathcal{M} \vDash A$ iff $\mathcal{N} \vDash A$.

'$\approx$' is clearly an equivalence relation. Now let $\mathcal{M} = \langle W,R,S \rangle$ be a model. For each $w \in W$, let $[\mathcal{M},w] = \{A\colon \mathcal{M}, w \vDash A\}$. Let $\mathbf{W} = \{\Gamma \subset \mathbf{FOR}\colon \Gamma = [\mathcal{M},w]$, for some $\mathcal{M}$, some $w\}$, and let now $\mathbf{S}$ be the class of all sets Σ, such that, for some formula A, Σ is A-saturated.

Now we can prove the following:

Lemma L2. If $[\mathcal{M},w] \vdash A$ *then* $\mathcal{M}, w \vDash A$.

Proof. If $A \in [\mathcal{M},w]$, then $\mathcal{M}, w \vDash A$ by definition. If A is a theorem, then A is valid, hence $\mathcal{M}, w \vDash A$. If A was obtained through uses of *MP* or *RK*, then $\mathcal{M}, w \vDash A$, because these rules (see proof of T10) are validity preserving. ■

Lemma L3. $\mathbf{W} = \mathbf{S}$.

[26] This also holds for **ZCG**.

[27] This also holds for **ZCP** and **ZP5**.

[28] This also holds for **ZCG** and **ZCP**.

Proof.

($\Rightarrow$) Let us suppose that $\Gamma \in \mathbf{W}$: thus, for some $\mathcal{M} = <W,R,S>$ some $w \in W$, $\Gamma = [\mathcal{M},w]$.

(i) First of all, $\mathcal{M},w \nvDash \neg(A \rightarrow A)$, for all wffs A, since $\mathcal{M},w \vDash A \rightarrow A$, and then, because of L2, $[\mathcal{M},w] \nvdash \neg(A \rightarrow A)$.

(ii) Now we have the following: for all B, if $B \notin \Gamma$, then $\Gamma \cup \{B\} \vdash \neg(A \rightarrow A)$, because $B \notin [\mathcal{M},w]$, hence $\mathcal{M},w \nvDash B$, $\mathcal{M},w \vDash \neg B$ and, since $\vdash \neg B \rightarrow (B \rightarrow \neg(A \rightarrow A))$, $\mathcal{M},w \vDash \neg B \rightarrow (B \rightarrow \neg(A \rightarrow A))$, $\neg B \rightarrow (B \rightarrow \neg(A \rightarrow A)) \in [\mathcal{M},w]$, $[\mathcal{M},w] \vdash \neg B \rightarrow (B \rightarrow \neg(A \rightarrow A))$, so $\Gamma \cup \{B\} \vdash \neg(A \rightarrow A)$.

From (i) and (ii), $[\mathcal{M},w]$ is a $\neg(A \rightarrow A)$-saturated set, for all A. Thus $[\mathcal{M},w] \in \mathbf{S}$, $\Gamma \in \mathbf{S}$.

($\Leftarrow$) $\Gamma \in \mathbf{S}$.

(a) We construct a model $\mathcal{M} = <\mathbf{S},\rho,\mu>$, such that $\mathcal{M},\Sigma \vDash A$ iff $A \in \Sigma$, for every wff A and every $\Sigma \in \mathbf{S}$. From P8.vi, we have that, if $\Gamma\rho\Sigma$ then $\Gamma\mu\Sigma$, so $\rho \subset \mu$. It is now easy to prove, using P8, that ρ and μ satisfy, for each logic, the required properties of the accessibility relations.

(b) We prove now the following: for every $\Sigma \in \mathbf{S}$, $\varepsilon(\Sigma^{\mathrm{B}}) = \cap\{\Theta \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{B}}) \subset \Theta\}$.

It is clear that $\varepsilon(\Sigma^{\mathrm{B}}) \subset \cap\{\Theta \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{B}}) \subset \Theta\}$. On the other direction, let A be such that $A \notin \varepsilon(\Sigma^{\mathrm{B}})$; then $\mathrm{B}A \notin \Sigma$. Since $\Sigma \in \mathbf{S}$, $\Sigma \nvdash \mathrm{B}A$, so by P7 there is an A-saturated set Σ^* such that $\varepsilon(\Sigma^{\mathrm{B}}) \subset \Sigma^*$. Then $A \notin \Sigma^*$, and $\Sigma^* \in \{\Sigma \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{B}}) \subset \Gamma\}$. From this it follows that $A \notin \cap\{\Theta \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{B}}) \subset \Theta\}$.

(c) In a similar way, for every $\Sigma \in \mathbf{S}$, $\varepsilon(\Sigma^{\mathrm{K}}) = \cap\{\Theta \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{K}}) \subset \Theta\}$.

(d) Hence, from (a) and (b) we have that $A \in \varepsilon(\Sigma^{\mathrm{B}})$ iff $A \in \cap\{\Theta \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{B}}) \subset \Theta\}$ iff $A \in \cap\{\Theta \in \mathbf{S}: \Sigma\rho\Theta\}$. From (a) and (c), $A \in \varepsilon(\Sigma^{\mathrm{K}})$ iff $A \in \cap\{\Theta \in \mathbf{S}: \varepsilon(\Sigma^{\mathrm{K}}) \subset \Theta\}$ iff $A \in \cap\{\Theta \in \mathbf{S}: \Sigma\mu\Theta\}$.

(e) We prove now that $\mathcal{M} = <\mathbf{S},\rho,\mu>$ is a Kripke model. $\mathbf{S}$ is a non-empty set, and $\rho \subset \mu \subset \mathbf{S} \times \mathbf{S}$. ρ and μ also have the desired properties. We show now that $\mathcal{M}$ fulfills the conditions of definition D2. For all $\Sigma \in \mathbf{S}$,

(i) $\mathcal{M},\Sigma \vDash \neg B$ iff $\neg B \in \Sigma$ iff $B \notin \Sigma$ iff $\mathcal{M},\Sigma \nvDash B$;

(ii) $\mathcal{M},\Sigma \vDash B \rightarrow C$ iff $B \rightarrow C \in \Sigma$ iff $B \notin \Sigma$ or $C \in \Sigma$ iff $\mathcal{M},\Sigma \nvDash B$ or $\mathcal{M},\Sigma \vDash C$;

(iii) $\mathcal{M},\Sigma \vDash \mathrm{K}B$ iff $\mathrm{K}B \in \Sigma$ iff $B \in \varepsilon(\Sigma^{\mathrm{K}})$ iff $B \in \cap\{\Theta \in \mathbf{S}: \Sigma\mu\Theta\}$ iff for all $\Theta \in \mathbf{S}$ such that $\Sigma\mu\Theta$, $B \in \Theta$;

(iv) $\mathcal{M},\Sigma \vDash \mathrm{B}B$ iff $\mathrm{B}B \in \Sigma$ iff $B \in \varepsilon(\Sigma^{\mathrm{B}})$ iff $B \in \cap\{\Theta \in \mathbf{S}: \Sigma\rho\Theta\}$ iff for all $\Theta \in \mathbf{S}$ such that $\Sigma\rho\Theta$, $B \in \Theta$.

(f) Now $\Gamma \in \mathbf{S}$, hence Γ is one of the worlds in $\mathcal{M}$. Now we define $\Delta_{\mathcal{M},\Gamma} = \{A: \mathcal{M},\Gamma \vDash A\}$. thus $[\mathcal{M},\Gamma] = \Gamma$, so $\Gamma \in \mathbf{W}$. ■

Theorem T11. (*Completeness*) $\Gamma \vdash A$ iff $\Gamma \vDash A$.

Proof. One direction is T10. To prove the other direction, let us suppose that $\Gamma \nvdash A$. From P6 there is an A-saturated set Σ such that $\Gamma \subset \Sigma$. From L2 there is a Kripke model $\mathcal{M} \in \mathbf{K}$ and w in $\mathcal{M}$ such that $\Sigma = [\mathcal{M},w]$. Thus for all $C \in \Gamma$, $\mathcal{M},w \vDash C$, and $\mathcal{M},w \nvDash A$. It thus follows that $\Gamma \nvDash A$. ■

Now that we have presented a semantics for the EDLs, and have proven correctness and completeness, it is time that we tie some loose ends from section 1.2, where we discusses modalities. Theorems T2 to T9 are stated in the form "there are *at most* ... modalities". What we should do now is to show that the numbers mentioned are exact. Thus:

Theorem T12. *In* **Z, ZG, ZC, ZCG, ZCP, ZP, Z5** *and* **ZP5**, *the number of distinct modalities is, respectively, 84, 46, 18, 18, 10, 14, 18 and 10.*

Proof. Since Theorems T2 through T9 state that the logics have at most the mentioned modalities, we need to show that one cannot reduce them further. For instance, we affirmed that, in **Z**, K and KBK are distinct modalities. Since K*A* → KBK*A* is a theorem of **Z**, we have to show that KBK*A* → K*A* doesn't hold (else they are equivalent). Consider the following model $\mathcal{M} = \langle\{x, y, z\}, \{\langle x,z\rangle, \langle y,z\rangle, \langle z,z\rangle\}, \{\langle x,y\rangle, \langle y,z\rangle, \langle x,z\rangle, \langle x,x\rangle, \langle y,y\rangle, \langle z,z\rangle\}\rangle$, such that $x(p) = 1$, $y(p) = 0$ and $z(p) = 1$. It is easy to check out that *R* is serial, transitive and euclidean; that *S* is reflexive and transitive; and that *R* is contained in *S*. It is also easy to see, in the next picture, that this model falsifies KBK*p* → K*p*.

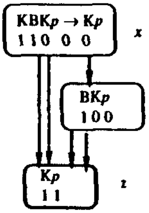

fig. 8

Here the smooth arrows represent the *S* relation, and the other ones, the *R* relation. (*zRz* was represented by a thicker outline, as you may notice. *S* reflexivity was left out.) Thus KBK*A* ↔ K*A* doesn't hold in **Z**. In a similar way, we have to show, for each pair of modalities, that they are not equivalent—what I won't do here for reasons of space. ■

2

Minimal Belief States

If you think the problem is bad now, just wait until we've solved it.
A. KASSPE

Clysterium donare,
Postea seignare,
Ensuita purgare.

MOLIÈRE, *Le Malade Imaginaire.*

Now that we are done with this overview of the EDL-systems, and are hopefully more in clear about the properties of the logics that we are using, we can move to considering our main problem, namely, how to characterize Angela's epistemic states, in the cases where she knows or believes only some formula α. We should maybe begin by asking what does this actually mean. By stating, for instance, that "Angela believes only α" we are surely *not* pretending to assert that *α is the only one proposition* Angela believes—just remember, she believes all tautologies, that is, ali tautologies, which are quite a lot, are already contained in her belief state. "believes only" could then be better understood as meaning that the formula α should be some kind of information sufficient to "reconstruct" or "characterize" or "determine" Angela's belief state; in other words, with α in our hands we should be able to know what is in Angela's belief state. α would be in this sense more a kind of "minimal description", or a "key". This naturally leads to the question of what kind of formula can α be? We don't really want to narrow our choices just to propositional variables: Angela can, for instance, only believe that "if p then q". Even if she doesn't believe either p or q, this situation is of course different of believing just tautologies, because "if p then q" really tells us something about the world. So we should allow α to be at least any propositional (i.e., zero-degree) formula whatsoever. But why exclude modalized ones? *Prima facie* there is nothing which speaks against them: some of them will certainly show themselves to be "dishonest" (the preferred example in [HM] is the formula $\alpha = \mathrm{K}p \vee \mathrm{K}q$), but others won't (like for instance $\mathrm{B}p$). So let us agree that α can be any formula of L, modalized or not.

We should next decide which kind of state we would actually like to characterize: a knowledge state, a belief state, or both of them? Well, in ali our systems we have the formula $\mathrm{K}A \to \mathrm{B}A$ as an axiom, and this means that Angela believes every proposition she knows. If we now consider epistemic states as being sets of formulas, then this would intuitively mean that a knowledge state is always a (probably proper) subset of a belief state. Thus belief states are more comprehensive—and since HM understand default rules anyway as "rules of conjecture", like $B(\neg Kp \rightarrow q)$, we could then concentrate mainly on belief states. An additional reason is that agents, in order to act, normally also take into account what they believe, not only what they know. So this should settle the question. In the following sections, then, we consider different ways of characterizing belief states.

2.1 Stable sets

A first tentative of characterizing belief states uses the notion of "stable sets", a denomination that was introduced by Stalnaker.[29] Of course, in the original discussion this notion referred only to "knowledge" sets, so we have to adapt it here. Well, in all EDLs the "belief branch" (i.e., the set of all formulas in which no K-operator occurs) is as strong as the modal calculus KD45.[30] For this reason I suggest for stable sets the following definition (essentially the same that was already used in [HM84] for knowledge stable sets, with the difference that we now work with beliefs):

Definition D3. Let L be any EDL-logic. A set S of formulas is an *L-stable* set if:

(st1) S is closed under logical consequences;

(st2) $A \in S$ iff $BA \in S$;

(st3) $A \notin S$ iff $\neg BA \in S$;

(st4) S is consistent.

This is a general definition, and can be used in every EDL-system, but of course each system will determine in its own way which formulas should belong to the stable set. I'll try to make this clear with an example. Suppose we have a situation in which Angela knows and believes p; believes, but doesn't know, q; and neither believes nor knows r. That is, we have:

$$Kp, Bp, Bq, \neg Kq, \neg Kr, \neg Br.$$

Now, in each of the different logics, Angela's belief state would look like the following (where 'bs_L' abbreviates the belief state in logic L):

bs_Z:	$\{ p, q, Kp, Bp, Bq, KKp, BBp, BBq, KBp, \neg Br, \ldots \}$		
bs_{ZP}:	bs_Z	$\cup$	$\{ KBq, K\neg Br, \ldots \}$;
bs_{ZC}:	bs_Z	$\cup$	$\{ Kq, KKq, KBq, BKq, \ldots \}$;
bs_{ZCP}:	bs_{ZP}	$\cup$	bs_{ZC};
bs_{ZS}:	bs_Z	$\cup$	$\{ K\neg Kq, K\neg Kr, \ldots \}$;
bs_{ZPS}:	bs_{ZS}	$\cup$	bs_{ZP}.

[29] Cf. [HM84], p4.

[30] Just to remember, KD45 is also known as "weak S5", that is, S5 with $\Box\alpha \rightarrow \Diamond\alpha$ instead of the reflexivity axiom $\Box\alpha \rightarrow \alpha$.

Supposing further that she doesn't know that she doesn't know r ($\neg$K$\neg$Kr), we would have:

bs_{ZG}: bs_Z $\cup$ { $\neg$K$\neg r$, ... };
bs_{ZCG}: bs_{ZC} $\cup$ { $\neg$K$\neg r$, ... }.

In this example I have included in the bs_L-sets only the formulas that should necessarily be on them. If we had for instance that Angela also knows that she believes q, then KBq would of course also belong to bs_Z. But in Z Bq→KBq is not a theorem, so KBq would hold for another reasons, and we could entirely as well have a different situation in which KBq is not true. This is in Z still possible. But KBq must be in bs_{ZP}, because Bq→KBq is an axiom of ZP, hence believed by ideal user Angela. Thus there are no ZP-stable sets containing Bq and not containing KBq as well. Similar holds of the other logics. Notice also that the additional hypothesis concerning the logics using axiom g wouldn't hold in Z5 and ZP5, because in this logics Angela is fully instrospective, hence she knows when she doesn't know something.

But let us proceed. Suppose now that Angela believes only the formula α (which can be of course a conjunction of other formulas). How can we characterize Angela's belief state? It is clear that this state must contain α—but we have obviously a lot of states to which α belongs. This particular belief state should then be the "minimal", whatever we choose here "minimal" to mean. The easiest and most elegant solution would be to use the notion of set inclusion: let us take all stables sets containing α, and the smallest of them is now Angela's belief state when she believes only α.

But nothing in life is that easy, as we can see with the next proposition.

Proposition P9. [HM84] *No stable set is a proper subset of another stable set.*

Proof. As in [HM84], p. 5: suppose there is two stable sets S and T such that S ⊂ T. Hence there is $A \in$ S, $A \notin$ T. From definition D3 we have B$A \in$ S and $\neg$B$A \in$ T. But B$A \in$ S implies B$A \in$ T, in which case T would be inconsistent, and this cannot be, by definition. ■

Oh well, there must be other ways of killing this cat. HM's solution goes as follows:

> A possible candidate for the 'minimal' [belief] state containing α is the stable set containing α whose *propositional* subset[31] is minimum (w.r.t inclusion). ([HM84], p. 5, italics mine.)

That this solution works in the pure knowledge logic arises from the fact that in [HM84] a stable set is *uniquely determined* by its propositional subformulas. But here this is not always the case, as we can see on the following example.[32] Let us suppose that we have two different situations (call them **a** and **b**) such that in **a** Angela knows that p, but doesn't know that q; and the other way round in **b**. In both cases she believes that p and that q. So:

a = { Kp, Bp, $\neg$Kq, Bq, ...}

[31] Under "propositional subset" should be understood the set of all formulas in which no K- or B-operator occurs—or whose modal degree is zero (see definition in Chapter 1).

[32] With exception of the systems ZC, ZCP, ZCG and, in certain aspects, of Z5 and ZP5.

$\mathbf{b} = \{\neg Kp, Bp, Kq, Bq, ...\}$

The corresponding (Z–)stable sets would then be:

$$bs_{\mathbf{a}} = \{\ p, q, Kp, - , Bp, Bq, BKp, - , ...\ \}$$
$$bs_{\mathbf{b}} = \{\ p, q, - , Kq, Bp, Bq, - , BKq, ...\ \}$$

As we can see, $bs_{\mathbf{a}} \neq bs_{\mathbf{b}}$, even if their propositional subsets are the same, namely the set $\{p, q\}$. In spite of Angela's believing the same "facts" (p and q) in *both* situations, what she believes about her own internal states is different in each of them. I'd like to remark here that this is only so because the known facts are different in the two worlds, that is, because the "pure knowledge" propositional subset (zero-degree wffs that Angela knows) is not the same—that's the reason for our problem. (Remember we couldn't reduce formulas to first-one degree ones? Here is where we are going to miss that.) So we are bound to run into trouble with some logics. But before we dive into these waters, let us examine closer the cases in which things work. What we will be trying to do is to find under which conditions two stable sets are the same, i.e., which kind of subsets uniquely determine a stable set. If we have this, we can, as it will be shown later, define a kind of smaller-than relation, and then identify the minimal belief state we are looking for.

1. The ZC/ZCG/ZCP *solution*

In ZCP we can easily prove that $BA \leftrightarrow \neg K\neg KA$ is derivable, or equivalently, that one can define the operator 'B' in terms of 'K' (cf. Appendix A3). We can hence consider ZCP, in the end, as being a pure knowledge logic, as strong as the modal system S4.2. With this ZCP loses some of its interest to us, because the characterization problem reduces itself to the level of the pure knowledge logic. Anyway, we can find for it, and for ZC/ZCG as well, a method of characterizing minimal belief states. Since *c* ($BA \rightarrow BKA$) is an axiom of these systems, we have $B(BA \rightarrow KA)$ as theorem (see Appendix A4). (Just to remember, in these calculi we should better interpret 'B' as conviction.) A natural language rendering of this formula could be: Angela is convinced that, when she is convinced of A, then she knows that A. Now this interestingly means that, in Angela's belief states, conviction and knowledge are equivalent. It is easy to see why: $\vdash B(BA \rightarrow KA)$ entails that $BA \rightarrow KA$ belongs to the belief state. This is also the case for $KA \rightarrow BA$, which it is an axiom. Hence $KA \leftrightarrow BA$ will also be in Angela's belief state, and the direct consequence of this is that ZC/ZCG/ZCP–stable sets have the same characteristics of the knowledge stable sets in [HM84]. Let us call these κ-*stable sets*, and they are defined as follows:

Definition D4. A set S of formulas is a κ-*stable* set iff:

(κ-st1) S is closed under logical consequences;

(κ-st2) $A \in S$ iff $KA \in S$;

(κ-st3) $A \notin S$ iff $\neg KA \in S$;

(κ-st4) S is consistent.

Proposition P10. *Let* S *be a* ZC/ZCG/ZCP*–stable set. Then* S *is* κ*-stable.*

Proof. κ-st1 and κ-st4 follow immediately from the definition of stable sets. The other two conditions follow as easily:

$$\begin{array}{llll}
(\kappa\text{-st2})\ A \in \mathrm{S} & \text{iff} & \mathrm{B}A \in \mathrm{S} & \text{(D3)} \\
 & \text{iff} & \mathrm{BK}A \in \mathrm{S} & (\vdash \mathrm{BK}A \leftrightarrow \mathrm{B}A) \\
 & \text{iff} & \mathrm{K}A \in \mathrm{S} & \text{(D3).} \\
 & & & \\
(\kappa\text{-st3})\ A \notin \mathrm{S} & \text{iff} & \neg\mathrm{B}A \in \mathrm{S} & \text{(D3)} \\
 & \text{iff} & \mathrm{B}\neg\mathrm{K}A \in \mathrm{S} & (\vdash \neg\mathrm{B}A \leftrightarrow \mathrm{B}\neg\mathrm{K}A) \\
 & \text{iff} & \neg\mathrm{K}A \in \mathrm{S} & \text{(D3).} \blacksquare
\end{array}$$

As a consequence of this proposition we can consider ZC/ZCG/ZCP–stable sets as sets in which one reasons with the rules of the pure knowledge logic. But the most interesting in this story is the fact that κ-stable sets are uniquely determined by their propositional subsets—what was already proved in [HM84] (p. 4, Proposition 1: result due to Moore [Mo83]). To prove it here we need the following definitions. Where S is a stable set, we say that $pr_B(\mathrm{S}) = \{A \in \mathrm{S}: dg(A) = 0\}$ is the *propositional belief subset* of S.

Lemma L4. ([HM84] , Lemma 1) *Let* S *be a* ZC/ZCG/ZCP*–stable set. Then:*

(a) $\mathrm{K}A \vee B \in \mathrm{S}$ *iff* $A \in \mathrm{S}$ *or* $B \in \mathrm{S}$;

(b) $\neg\mathrm{K}A \vee B \in \mathrm{S}$ *iff* $A \notin \mathrm{S}$ *or* $B \in \mathrm{S}$.

Proof. Exactly like in [HM84]—or similar to the one of Lemma L5 below.

The next theorem establishes then that ZC/ZCG/ZCP–stable sets are uniquely determined by their pr_B-subsets.

Theorem T13. *Let* S *and* T *be* ZC/ZC2/ZCP*–stable sets such that* $pr_B(\mathrm{S}) = pr_B(\mathrm{T})$. *Then* S = T.

Proof.[33] If A is a formula, let A^* be a formula obtained from A where all B-operators were replaced by K. It is first of all easy to prove that there is for S and T corresponding sets S* and T*, such that $A \in \mathrm{S}$ (T) iff $A^* \in \mathrm{S}^*$ (T*). It is also easy to prove that S* und T* are closed under S5-consequences. We then prove that $\mathrm{S}^* = \mathrm{T}^*$, that is, for ali formulas B, $B \in \mathrm{S}^*$ iff $B \in \mathrm{T}^*$. If $dg(B) = 0$, we don't need to prove anything, since S and T agree on propositional formulas, and hence S* and T* too. Let us then suppose that $dg(B) = 1$. Since both sets are closed under S5-consequences, B is equivalent to a first degree conjunction B^* of disjunctions D_i such that, for each D_i, $D_i = \mathrm{K}C_1 \vee \ldots \vee \mathrm{K}C_m \vee \neg\mathrm{K}C_{m+1} \vee \ldots \vee \neg\mathrm{K}C_k \vee E$, where E is a propositional formula, the same as each C_j, since $dg(B^*) = 1$. Now $B^* \in \mathrm{S}^*$ iff each conjunct $D_i \in \mathrm{S}^*$, and this holds, by Lemma L4, iff one of the following holds: $C_1 \in \mathrm{S}^*, \ldots, C_m \in \mathrm{S}^*$, $C_{m+1} \notin \mathrm{S}^*, \ldots, C_k \notin \mathrm{S}^*$, $E \in \mathrm{S}^*$. A similar property holds for T*, and, since S* and T* agree on propositional formulas, $B^* \in \mathrm{S}^*$ iff $B^* \in \mathrm{T}^*$, $B \in \mathrm{S}^*$ iff $B \in \mathrm{T}^*$. Hence $A \in \mathrm{S}$ iff $A \in \mathrm{T}$. ■

[33] The proof of this theorem, as well as of T14 and T15, is adapted from [HM84], Proposition 1, p. 4.

The consequence of this all is that now we have, when we are working with **ZC**, **ZCG** and **ZCP**, a method of characterizing Angela's belief state when she believes only some formula α—it will be the stable set whose pr_B-set is the "smallest"—in a sense of "small" we are going to define later on. Let us first look at the other logics.

2. *The* ZP5 *solution*

Another among the above mentioned nice cases—and this only with some restrictions—is **ZP5**, one of the strongest EDL-systems. To prove that **ZP5**-stable sets are uniquely determined by their propositional (belief and *also knowledge*) subsets we need first some definitions and lemmas. If S is a stable set, we say that $pr_K(\mathrm{S}) = \{A \in \mathrm{S}: dg(A) = 0 \wedge \mathrm{K}A \in \mathrm{S}\}$ is its *propositional knowledge subset*. Of course $pr_K(\mathrm{S}) \subseteq pr_B(\mathrm{S})$. As we'll see, the pr_K-sets must also play a role.

Lemma L5. *Let* S *be a* **ZP5**-*stable set. Then:*

(a) $\mathrm{B}A \vee B \in \mathrm{S}$ *iff* $A \in \mathrm{S}$ *or* $B \in \mathrm{S}$;

(b) $\neg\mathrm{B}A \vee B \in \mathrm{S}$ *iff* $A \notin \mathrm{S}$ *or* $B \in \mathrm{S}$;

(c) $\mathrm{K}A \vee B \in \mathrm{S}$ *iff* $A \in pr_K(\mathrm{S})$ *or* $B \in \mathrm{S}$, *if* $dg(A) = 0$;

(d) $\neg\mathrm{K}A \vee B \in \mathrm{S}$ *iff* $A \notin pr_K(\mathrm{S})$ *or* $B \in \mathrm{S}$, *if* $dg(A) = 0$.

Proof. We prove the cases (c) and (d).

(c$\Rightarrow$) $\mathrm{K}A \vee B \in \mathrm{S}$. Thus $\mathrm{K}A \in \mathrm{S}$ or $B \in \mathrm{S}$. If $B \in \mathrm{S}$ we are done, so let us consider $\mathrm{K}A \in \mathrm{S}$. From axiom *k* it follows that $A \in \mathrm{S}$ and, since $dg(A) = 0$, that $A \in pr_K(\mathrm{S})$.

(c$\Leftarrow$) $A \in pr_K(\mathrm{S})$ or $B \in \mathrm{S}$. If $B \in \mathrm{S}$ then $\mathrm{K}A \vee B \in \mathrm{S}$. If $A \in pr_K(\mathrm{S})$ then by definition $\mathrm{K}A \in \mathrm{S}$, and $\mathrm{K}A \vee B \in \mathrm{S}$.

(d$\Rightarrow$) $\neg\mathrm{K}A \vee B \in \mathrm{S}$. If $A \in pr_K(\mathrm{S})$, by definition $\mathrm{K}A \in \mathrm{S}$, and it follows from the hypothesis that $B \in \mathrm{S}$. If $A \notin pr_K(\mathrm{S})$ then it's alright.

(d$\Leftarrow$) $A \notin pr_K(\mathrm{S})$ or $B \in \mathrm{S}$. If $B \in \mathrm{S}$ then $\neg\mathrm{K}A \vee B \in \mathrm{S}$ and we are finished. So let us suppose that $B \notin \mathrm{S}$ and $\neg\mathrm{K}A \vee B \notin \mathrm{S}$. Then $\neg\mathrm{K}A \notin \mathrm{S}$ and $B \notin \mathrm{S}$. If $\neg\mathrm{K}A \notin \mathrm{S}$ then $\neg\mathrm{B}\neg\mathrm{K}A \in \mathrm{S}$ (st4). Since $\vdash \neg\mathrm{K}A \rightarrow \mathrm{B}\neg\mathrm{K}A$, $\neg\neg\mathrm{K}A \in \mathrm{S}$, $\mathrm{K}A \in \mathrm{S}$ and, since $dg(A) = 0$, $A \in pr_K(\mathrm{S})$, which is a contradiction. Hence $\neg\mathrm{K}A \vee B \in \mathrm{S}$.

For (a) and (b) the proof is similar and even simpler. ■

The next theorem now shows that **ZP5**-stable set are uniquely determined by *both* their propositional subsets (pr_K and pr_B).

Theorem T14. *Let* S *and* T *be* **ZP5**-*stable sets, such that* $pr_K(\mathrm{S}) = pr_K(\mathrm{T})$, *and* $pr_B(\mathrm{S}) = pr_B(\mathrm{T})$. *Then* S = T.

Proof. We prove that for every formula A, $A \in \mathrm{S}$ iff $A \in \mathrm{T}$. By Propositions P1 and P2 we have that $A \leftrightarrow A^*$, where A^* is at most of the first degree and is in MCNF. If $dg(A^*) = 0$, we don't need to prove anything, since S and T agree on propositional formulas. Let us thus suppose that $dg(A^*) = 1$. Since it is in MCNF, A^* is a conjunction of disjunctions D_i, such that each $D_i \doteq \mathrm{K}B_1 \vee \ldots \vee \mathrm{K}B_m \vee \neg\mathrm{K}B_{m+1} \vee \ldots \vee$

$\neg KB_k \vee BC_1 \vee ... \vee BC_r \vee \neg BC_{r+1} \vee ... \vee \neg BC_s \vee E$, where E is a propositional formula, same as each B_j, C_j, since $dg(A^*) = 1$. Now $A^* \in$ S iff each conjunct $D_i \in$ S, and this holds, by Lemma L5, iff one of the following holds: $B_1 \in pr_K(S)$,..., $B_m \in pr_K(S)$, $B_{m+1} \notin pr_K(S)$,..., $B_k \notin pr_K(S)$, $C_1 \in$ S,..., $C_r \in$ S, $C_{r+1} \notin$ S,..., $C_s \notin$ S, $E \in$ S. A similar property holds for T, and, since S and T agree on propositional formulas, and since $pr_K(S) = pr_K(T)$, $A^* \in$ S iff $A^* \in$ T, $A \in$ S iff $A \in$ T. ■

We can see in this proof the importance of the restriction $pr_K(S) = pr_K(T)$: in order to show that $A \in$ S iff $A \in$ T we need to suppose that both sets agree on some "basic" formulas—in this case not only the *believed* zero-degree formulas, but also the *known* ones. This would seem to imply that, in order to characterize a minimal belief state, we not only need to know that α everything Angela believes, but also that some β is everything she knows. On the other hand, betwen a state in which she believes α and knows β, and another in which she believes α and knows nothing, the second is clearly the smaller—the smallest of all, thus, being the one in which Angelas has only beliefs and no knowledge.

3. The Z5 solution

With respect to this logic the situation is somewhat more complicated, but anyway not beyond salvation. In the first place, it is easy to see that stable sets are not uniquely determined by their propositional subsets *alone*. Let us imagine two different situations (call them again a and b, and let bs_a and bs_b be stable sets denoting Angela's belief state in each case) such that $pr_K(bs_a) = pr_K(bs_b)$, and $pr_B(bs_a) = pr_B(bs_b)$. Let us further suppose that in a Angela knows that she believes A (KBA), while in case b she doesn't ($\neg$KBA). We would consequently have K$BA \in bs_a$, $\neg$K$BA \in bs_b$, and of course $bs_a \neq bs_b$. We must hence introduce further restrictions in order to characterize the desired belief states. This is so, of course, because in **Z5** we can't reduce every formula to a first-degree one, only to the *second* degree. So we need one construction more, namely of a *BT-set*. For any stable set S, we define BT(S) as $\{A \in$ S: A = KB and B is a β^0-formula$\}$.

We can then go to the next theorem, which proves that **Z5**-stable sets are uniquely determined by their pr_K-, pr_B- and BT-sets.

Theorem T15. *Let* S *and* T *be* **Z5**-*stable sets, such that* $pr_K(S) = pr_K(T)$, $pr_B(S) = pr_B(T)$ *and* $BT(S) = BT(T)$. *Then* S = T.

Proof. Similar to theorem T14, with now the BT-sets also playing the important role. ■

4. The lack of a **Z**, **ZG** *and* **ZP** *solution*

Now these are the complicated cases—these logics are going to stay for the present as open problems, at least in what concerns the characterization of belief states through stable sets the way we are trying it here.

5. Defining a smaller-than relation

Now with respect to the other systems for which we had a solution, we are able to give for them a definition of "honesty" of a formula. We must first only define a kind of "smaller than" relation between stable sets, something corresponding to the notion of set inclusion.

In the logics ZC, ZCG and ZCP, this is no big problem. In Theorem T13 we have proved that ZC/ZCG/ZCP-stable sets are uniquely determined by their pr_B-subsets. So it is enough that $pr_B(S)$ is a proper subset of $pr_B(T)$ to characterize S as a state in which Angela believes less than in T. The following table show the possible relations that obtain between the pr_B-subsets of two stable sets S and T (where 'S $<_b$ T', or 'S $=_b$ T' mean that in S Angela believes less than, or the same as, in T):

if ...	*we have ...*
$pr_B(S) \subset pr_B(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$S =_b T$
$pr_B(S) \supset pr_B(T)$	$T >_b S$

This could lead us then to the following definition:

Definition D5a. Let S and T be stable sets. We say that S is ZC/ZCG/ZCP-*smaller* than T iff $pr_B(S) \subset pr_B(T)$.

In ZP5 things aren't that easy, because here we have to consider the two propositional subsets of a stable set (cf. T14). Let us examine the possible configurations:

if ...	*and ...*	*we have ...*
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$S <_b T$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	??
$pr_B(S) \subset pr_B(T)$	$pr_K(S) = pr_K(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$pr_K(S) = pr_K(T)$	$S =_b T$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) = pr_K(T)$	$T >_b S$
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	??
$pr_B(S) = pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$T =_b S$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$T >_b S$

As one can see, in three of the lines S is smaller than T; in other three T is smaller than S; in one they are the same and, in two lines (marked with '??'), there is no comparison possible. We arrive then to the following definition:

Definition D5b. Let S and T be stable sets. We say that S is ZP5-*smaller* than T iff $pr_B(S) \subset pr_B(T)$ and $pr_K(S) \subseteq pr_K(T)$, or if $pr_B(S) = pr_B(T)$ and $pr_K(S) \subset pr_K(T)$.

In the case of **Z5**, now, things are going to get really tough, because here we have to consider three different subsets of a stable set (cf. T15). Let us try to make some sense of all the possible combination in the following table:

if ...	*and ...*	*and ...*	*we have ...*
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) \subset BT(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) \subset BT(T)$	$S <_b T$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) \subset BT(T)$	??
$pr_B(S) \subset pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) \subset BT(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) \subset BT(T)$	$S <_b T$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) \subset BT(T)$	??
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) \subset BT(T)$	??
$pr_B(S) = pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) \subset BT(T)$	??
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) \subset BT(T)$	??
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) = BT(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) = BT(T)$	$S <_b T$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) = BT(T)$	??
$pr_B(S) \subset pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) = BT(T)$	$S <_b T$
$pr_B(S) = pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) = BT(T)$	$S =_b T$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) = BT(T)$	$T <_b S$
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) = BT(T)$	??
$pr_B(S) = pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) = BT(T)$	$T <_b S$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) = BT(T)$	$T <_b S$
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) \supset BT(T)$	??
$pr_B(S) = pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) \supset BT(T)$	??
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \subset pr_K(T)$	$BT(S) \supset BT(T)$	??
$pr_B(S) \subset pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) \supset BT(T)$	??
$pr_B(S) = pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) \supset BT(T)$	$T <_b S$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) = pr_K(T)$	$BT(S) \supset BT(T)$	$T <_b S$
$pr_B(S) \subset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) \supset BT(T)$	??
$pr_B(S) = pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) \supset BT(T)$	$T <_b S$
$pr_B(S) \supset pr_B(T)$	$pr_K(S) \supset pr_K(T)$	$BT(S) \supset BT(T)$	$T <_b S$

One can see that we have more undecided cases as in the logic before. Anyway, summing up what this table tells us, we arrive at the following

Definition D5c. Let S and T be stable sets. Then S is **Z5**-*smaller* than T iff (i) $pr_B(S) \subset pr_B(T)$, $pr_K(S) \subseteq pr_K(T)$ and $BT(S) \subseteq BT(T)$; or (ii) $pr_B(S) = pr_B(T)$, $pr_K(S) \subset pr_K(T)$ and $BT(S) \subseteq BT(T)$; or (iii) $pr_B(S) = pr_B(T)$, $pr_K(S) = pr_K(T)$ and $BT(S) \subset BT(T)$.

We can now characterize Angela's belief state, in which she believes only α, as the "L-smallest" stable set containing α, for $L \in \{$**ZC**, **ZCG**, **ZCP**, **Z5**, **ZP5**$\}$. There are of course lots of formulas α for which there is no such a state, for instance let $\alpha = \mathrm{B}p \vee \mathrm{B}q$. This leads us to the following definition of honesty: for $L \in \{\mathbf{ZC}, \mathbf{ZCG}, \mathbf{ZCP}, \mathbf{Z5}, \mathbf{ZP5}\}$, a formula α is L-*honest*$_S$ iff there is an L-smallest stable set S containing α.

We can see at once that the formula $\alpha = \mathrm{B}p \vee \mathrm{B}q$ is not **ZC/ZCG/ZCP**-honest$_S$. All **ZC/ZCG/ZCP**-stable sets which contain α must also contain either p or q. Further, there is a **ZC/ZCG/ZCP**-stable set S_p which contains α and p, but not q, and another set S_q which contains α and q, but not p. Neither S_p nor S_q are **ZC/ZCG/ZCP**-smallest, and the intersection $\mathrm{S}_p \cap \mathrm{S}_q$ contains neither p nor q. Hence there is no **ZC/ZCG/ZCP**-smallest stable set T containing α, such that $pr_B(\mathrm{T}) \subset pr_B(\mathrm{S}_p)$ and $pr_B(\mathrm{T}) \subset pr_B(\mathrm{S}_q)$. Thus α is not **ZC/ZCG/ZCP**-honest$_S$. In a similar way we can show that α is not honest$_S$ in other systems as well.

2.2 A sidestep: saturated sets

An alternative way of characterizing Angela's knowledge state using these smaller-than relations just defined concerns saturated sets. We begin by establishing some relations between stable and saturated sets. The reader has surely noticed that a saturated set can be seen as a world—or, to put it better, as a world description: this description tells us what is true in the world, also including what Angela knows or believes—these are facts, too. So the following should be true: to each saturated set (world) Σ corresponds a stable set, namely the set of the formulas believed (in this world) by Angela, and this set is no other than $\varepsilon(\Sigma^{\mathrm{B}})$. As we prove in the next two propositions:

Proposition P11. *Let* Σ *be a C-saturated set. Then* $\varepsilon(\Sigma^{\mathrm{B}})$ *is a stable set.*[34]

Proof. We prove that $\varepsilon(\Sigma^{\mathrm{B}})$ fulfills the conditions of definition D3.

(st1) Let A be a PC-tautology; thus $\vdash A$, $\vdash \mathrm{B}A$ (by *RB*), so $\Sigma \vdash \mathrm{B}A$ and $\mathrm{B}A \in \Sigma$, $A \in \varepsilon(\Sigma^{\mathrm{B}})$. Let us now suppose that $A, A \rightarrow B \in \varepsilon(\Sigma^{\mathrm{B}})$. Thus we have $\mathrm{B}A$, $\mathrm{B}(A \rightarrow B) \in \Sigma$. From k^b and *MP* it follows that $\mathrm{B}B \in \Sigma$ and finally $B \in \varepsilon(\Sigma^{\mathrm{B}})$.

(st2) Let us suppose that $A \in \varepsilon(\Sigma^{\mathrm{B}})$. So $\mathrm{B}A \in \Sigma$ and, since we have $\mathrm{B}A \rightarrow \mathrm{BB}A$ as an axiom, $\mathrm{BB}A \in \Sigma$, $\mathrm{B}A \in \varepsilon(\Sigma^{\mathrm{B}})$. On the other direction, if $\mathrm{B}A \in \varepsilon(\Sigma^{\mathrm{B}})$, then $\mathrm{BB}A \in \Sigma$. But $\vdash \mathrm{BB}A \rightarrow \mathrm{B}A$, so $\mathrm{B}A \in \Sigma$, $A \in \varepsilon(\Sigma^{\mathrm{B}})$.

(st3) If $A \notin \varepsilon(\Sigma^{\mathrm{B}})$ then $\mathrm{B}A \notin \Sigma$, $\neg\mathrm{B}A \in \Sigma$ and, from 5^b, $\mathrm{B}\neg\mathrm{B}A \in \Sigma$, $\neg\mathrm{B}A \in \varepsilon(\Sigma^{\mathrm{B}})$. On the other direction let us suppose that $\neg\mathrm{B}A \in \varepsilon(\Sigma^{\mathrm{B}})$, $A \in \varepsilon(\Sigma^{\mathrm{B}})$. From $\neg\mathrm{B}A \in \varepsilon(\Sigma^{\mathrm{B}})$ we get $\mathrm{B}A \notin \varepsilon(\Sigma^{\mathrm{B}})$, $\mathrm{BB}A \notin \Sigma$. From $A \in \varepsilon(\Sigma^{\mathrm{B}})$ it follows that $\mathrm{B}A \in \Sigma$, and, through 4^b, $\mathrm{BB}A \in \Sigma$—a contradiction. Thus $A \notin \varepsilon(\Sigma^{\mathrm{B}})$.

(st4) To prove that $\varepsilon(\Sigma^{\mathrm{B}})$ is consistent we have, since Σ is C-saturated, that for some wff C, it holds $\Sigma \nvdash C$, so Σ is consistent. Let us now suppose $\varepsilon(\Sigma^{\mathrm{B}})$ is inconsistent. Then there is an A such that A and $\neg A \in$

[34] When I talk about "stable" sets without specifying some EDL-system I am of course meaning that what is being said holds for *all* systems we are considering here.

$\varepsilon(\Sigma^B)$. From this fact it follows that BA, B$\neg A \in \Sigma$. But B$A \rightarrow \neg$B$\neg A$ is an axiom, and thus $\neg$B$\neg A \in \Sigma$, and Σ is in this case inconsistent, what cannot be. Hence $\varepsilon(\Sigma^B)$ is consistent. ■

Proposition P12. *Let* T *be a stable set. Then, for some C-saturated set* Σ, T = $\varepsilon(\Sigma^B)$.

Proof. Since T is stable, we know that T is consistent, so there is a formula C such that T $\nvdash C$. From proposition P6 it follows that there is a C-saturated set Σ, such that T $\subseteq \Sigma$. We have now to prove that T = $\varepsilon(\Sigma^B)$.

(A) Let us suppose, for some A, that $A \in$ T. Then B$A \in$ T (st2) and, since T $\subseteq \Sigma$, B$A \in \Sigma$, $A \in \varepsilon(\Sigma^B)$.

(B) Let us now have $A \in \varepsilon(\Sigma^B)$, $A \notin$ T. Then B$A \in \Sigma$. However, if $A \notin$ T, then $\neg$B$A \in$ T, $\neg$B$A \in \Sigma$, and this is a contradiciton. Hence $A \in$ T. ■

By now, it jumps to the eyes, since stable sets are the *bs*-sets of some saturated set, that there is (sort of) a way of defining honesty using saturated sets: α is L-honest$_W$ iff there is a A-saturated set Σ containing Bα such that $\varepsilon(\Sigma^B)$ is L-smallest (for $L \in$ {ZC, ZCG, ZCP, Z5, ZP5}). This is of course just another way of making use of stable sets.

2.3 Kripke models

A second method employed by HM in the characterization of knowledge states uses Kripke, or possible-world, models. Basically, the procedure is:

(i) define, for each model $\mathcal{M}$, the set of known formulas in $\mathcal{M}$ (namely the wffs that are true in every state of the model);

(ii) show that this set of known formulas is a stable set;

(iii) show that the model in which Angela knows only α is the union of ali models in which she knows α.

Well, in [HM84] this task is easily accomplished, and again this is so because the knowledge logic they used is S5. In models for this system, the accessibility relation must be an equivalence relation. Now this *almost* amounts to saying that each world is accessible to every world, which fact has as a consequence that one can completely delete the accessiblity relation from the picture: thus KA is true in a model if A is true in every world of the model.

But I said *almost*: in fact, we could have a model like the one in the following picture:

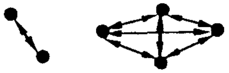

fig. 9

As one can see, the accessibility relation (depicted by arrows and black-filled circles in case of reflexivity) is an equivalence relation: it is easy to check visually that it is reflexive, symmetric and transitive. However, not every world is accessible to every world: they are grouped in different "clusters" which have no communication to another. As Hughes and Cresswell already pointed out ([HC72], p. 67), this means the same as having two S5-models glued together: since no cluster has any influence on the other, to evaluate valid formulas we have to get their values separatedly in each cluster—same procedure as looking in two different models. Hence we can in fact use models without the accessibility relation for S5—which we can call *monoclustered models.*

Now this has another interesting consequence: HM define the set $K(\mathcal{M})$ of the known facts in model $\mathcal{M}$ as $\{A : \mathcal{M},t \models A$, for every t in $\mathcal{M}\}$ (cf. [HM84], p. 7). It is now easy to show that

(1) $A \in K(\mathcal{M})$ iff $\mathcal{M},t \models$ KA for all t; and

(2) $A \notin K(\mathcal{M})$ iff $\mathcal{M},t \models \neg$K$A$ for all t.

(I) would give no problem even with standard (multiclustered) models, but (2) would. In a monoclustered model, if $A \notin K(\mathcal{M})$ then, for some state w, $\mathcal{M},w \not\models$ KA. So there is a state v such that $\mathcal{M},v \not\models A$. Since now every world is accessible to every world, for every t there is a world (namely v) where A is false, so KA is false in every world, and $\neg$KA is true in every world. That this doesn't work in a multiclustered model can be seen in the next picture:

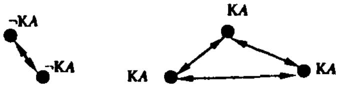

fig. 10

Here we have on the left a cluster where, for every t, $\mathcal{M},t \models \neg$K$A$. But worlds of this cluster are not accessible to worlds of the second one, so there, on the right, we have $\mathcal{M}, w \models$ KA, for every w. So $K(\mathcal{M})$ won't have the nice property (2), and won't be stable.

Now getting rid of this problem is only the first advantage of working with monoclustered models. The second concerns the method of characterizing the model in which Angela knows only α. As I said, a formula Kα is then true in a state s if and only if α is true in all states *simpliciter*. Now the intuition behind Kripke models is the following one: states are worlds which Angela thinks are possible relatively to what she knows/believes. If now a model $\mathcal{M}$ contains more states than another model $\mathcal{N}$, we can say that in $\mathcal{M}$ Angela is more ignorant than in $\mathcal{N}$. This fact implies that the model in which Angela knows only α should be the union $\mathcal{M}_\alpha$ of all models $\mathcal{M}$ such that $\mathcal{M} \models$ Kα, i.e., all models in which Kα holds. And this works with monoclustered models because we can take any two models whatsoever and nevertheless still be sure that their union will be a model. Bad luck, with our EDLs this is not always the case. Let us consider the following example: let $\mathcal{M} = \langle M,R_M,S_M \rangle$ and $\mathcal{N} = \langle N,R_N,S_N \rangle$ be two EDL-Kripke models, where $M = \{a, b\}$, $R_M = \{\langle a,b \rangle, \langle b,b \rangle\}$, $S_M = \{\langle a,a \rangle, \langle a,b \rangle, \langle b,b \rangle\}$, $N = \{a, c\}$, $R_N = \{\langle a,c \rangle, \langle c,c \rangle\}$, $S_N = \{\langle a,a \rangle, \langle a,c \rangle, \langle c,c \rangle\}$. Let now $\mathcal{U}$ the union of $\mathcal{M}$ and $\mathcal{N}$, that is, $\mathcal{U} = \langle U,R_U,S_U \rangle$, where $U = M \cup N$, $R_U = R_M \cup R_N = \{\langle a,b \rangle, \langle b,b \rangle, \langle a,c \rangle, \langle c,c \rangle\}$, $S_U = S_M \cup S_N$. The trouble here is that the belief accessibility relation R_U in $\mathcal{U}$ is not euclidean: we have aR_Ub and aR_Uc, but the pair $<b,c>$ doesn't belong to R_U. Thus $\mathcal{U}$ is not a Kripke model.

On the other hand there are some cases where the union of two models is still a model, namely, if the two original models don't have any states in common. (States, remember, are considered here to be truth-value assignments to propositional variables.) As follows:

Proposition P13. *Let $\mathcal{M} = <M,R_M,S_M>$ and $\mathcal{N} = <N,R_N,S_N>$ be two Kripke models such that $M \cap N = \emptyset$. Then $\mathcal{U} = <U,R_U,S_U>$, where $U = M \cup N$, $R_U = R_M \cup R_N$ and $S_U = S_M \cup S_N$, is a Kripke model.*

Proof. U is obviously a non-empty set. What we must show is that the relations R_U and S_U have the desired properties.

(a) R_U is serial, i.e., for every u in U there is a v such that uR_Uv. This is evident, because R_M and R_N are serial, and the pairs $<u,v>$ are consequently in R_U.

(b) R_U is transitive. Suppose not: then there is u, v, w in U such that uR_Uv, vR_Uw, but not uR_Uw. However, since $M \cap N = \emptyset$, we have as a consequence either (i) u, v and w are in M, in which case uR_Mv, vR_Mw and—since R_M is transitive—uR_Mw, with the consequence that uR_Uw; or (ii) u, v and w are in N, in hich case the same holds. So R_U is transitive.

(c) R_U is euclidean. Suppose not: then there is u, v, w in U, such that uR_Uv, uR_Uw, but not vR_Uw or wR_Uv. However, since $M \cap N = \emptyset$, we have as a consequence either (i) u, v and w are in M, in which case uR_Mv, uR_Mw and—since R_M is euclidean—vR_Mw, and hence vR_Uw; or (ii) u, v and w are in N, in which case the same holds. Hence R_U is euclidean.

In a similar way we can show that S_U has the desired properties. ■

How this fact could help us is still not clear to me. So how can we go on? Well, there should be a way of getting a kind of monoclustered model for knowledge and belief together. Let us see.

To begin with, in handling belief, things are likely to be somewhat different from the knowledge case. In fact, it is perfectly possible to have $\mathcal{M} \models$ BA (i.e., BA is true in every state $w \in M$) and nevertheless there could be $w^* \in M$ such that $\mathcal{M}, w^* \not\models A$. Now, if this happens, then w^* must be a special kind of world. If for instance there were a $t \in W$ such that tRw^*, then we would have $\mathcal{M},t \not\models$ BA (because there would be an accessible world with A false). So we can conclude the following: if $\mathcal{M} \models$ BA and there is w^* such that $\mathcal{M}, w^* \not\models A$, then there is no $t \in W$ such that tRw^*. If this is so, we say that w^* is a *lost world* (or *closed*, or *forbidden*—take your choice). Worlds that are not lost we will call *open*, or *accessible*.

The interesting about lost and open worlds seems to be that for KD45 (which is our belief logic here), we can put the open worlds together in the same basket: in fact, they are all accessible to every other world, if in the same cluster. A typical, multiclustered KD45-model could look like this:

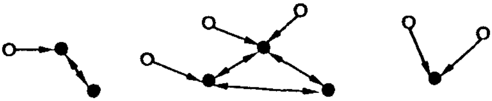

fig. 11

Unfilled circles represent the worlds that are not accessible to others, not even to themselves. The only thing we need to do, if we drop the belief accessibility relation from the picture, is to single out lost and open worlds when we define a model. So let us put all these ideas together and see what we get.

We define a monoclustered EDL-model as follows:

Definition D6. A *monoclusterd model* $\mathcal{M}$ is a pair $<W,O>$, where:

a. $W \neq \emptyset$;

b. each $w_i \in W$ is an assignment of truth-values to atomic formulas;

c. $O \subset W$ and $O \neq \emptyset$.

Again the elements of the set W of worlds are assignments of truth-values to propositional variables. O, of which we require to be non-empty, is the subset of W which contains the open worlds. Of course, the set W_L of lost worlds can be defined as $W-O$.

It should be now obvious, since we dropped the S-accessibility relation from the picture, that the knowledge branch of this as yet unknown EDL is **S5**. But do we really get **KD45** as belief logic? And which of our EDLs here is characterized by the class of monoclustered models?

Probably **ZP5**, and we'll see that this is the case. A first way to show that is to define for monoclustered models two accessibility relations over W, R^m and S^m, and to prove that they have exactly the characteristics of **ZP5** relations. So let $\mathcal{M} = <W,O>$ be a monoclustered model. For any two worlds w, $v \in W$, we say that wR^mv, if $v \in O$; and that wS^mv. Now we show that:

Lemma L6.

(i) *R^m is serial, transitive and euclidean.*

(ii) *S^m is an equivalence relation.*

(iii) $R^m \subset S^m$.

(iv) *1-mixed transitivity holds.*

Proof. (i) Since by definition O is non-empty, for every $w \in W$ there is a v such that wR^mv. So R^m is serial. Suppose now wR^mv and vR^mt. So $t \in O$, hence wR^mt, and R^m is transitive. Suppose now wR^mv and wR^mt. So $t \in O$, hence vR^mt, and R^m is euclidean.

(ii) Since wS^mv, for any two worlds w and v, S^m is obviously an equivalence relation.

(iii) Since $O \subset W$, it is trivial that $R^m \subset S^m$.

(iv) Suppose now wS^mv and vR^mt. So $t \in O$, hence wR^mt, and *1-mixed* transitivity holds. ■

It is also easy to see that, for instance, 2-mixed transitivity does not hold. Suppose wR^mv and vS^mt. So $v \in O$; however, we have no guarantee that t also belongs to O—it could be a lost world.

So we just got monoclustered models for at least one logic. Is there any chance of having this kind of model for the other systems as well? Well, the way we defined things entails that all these models validate the schema p. In view of this, it seems that if we want the models having this monoclusterdness characteristic—which is important in order to have BA true in the model iff A is true in all worlds—then we must accept that B$A \rightarrow$KBA shall turn out valid. Else there should be a world where BA is false; and yet another (obviously not in the same cluster, then) where A would be false. Thus BA would come out false in the model. Well, what we want to have is the following: the worlds that matter to evaluate *belief* formulas should be kept together in a cluster—this doesn't mean that worlds that matter for the cvaluation of *knowledge* formulas couldn't be arranged in a different way. To put it in another way, we could try models with the open–worlds–story instead of the belief accessibility relation, but introducing back again the S accessibility relation for knowledge.[35] Let us see what we get.

Definition D7. A *mixed monoclustered model* $\mathcal{M}$ is a triple $<W,O,S>$, where:

a. $W \neq \emptyset$;

b. each $w_i \in W$ is an assignment of truth-values to atomic formulas;

c. $O \subseteq W$ and $O \neq \emptyset$;

d. S is a binary relation over $W \times W$ which is at least reflexive and transitive; moreover, if $v \in O$ then for every w, wSv.

I claim that these just defined models are models for **ZP**. If we now add the following requirement:

e. for every $w, v \in W$, if $w \in O$ and wSv then $v \in O$;

then we should get models for **ZCP**.

We'll be proving all that soon. We must now redefine, for each formula A, what it means for A to be true in a (monoclustered) model and in a state. In what follows, I will use the following notation to denote monoclustered or mixed monoclustered models: $\mathcal{M} = <W,O[,S]>$. This just means that the relation S only applies, obviously, in the cases of **ZP** and **ZCP**.

Definition D8. Let $\mathcal{M} = <W,O[,S]>$ be a monoclustered model, and w an element of W:

a. $\mathcal{M}, w \models^m A$ iff $w(A) = 1$, if A is a propositional variable;

b. $\mathcal{M}, w \models^m \neg A$ iff $\mathcal{M}, w \not\models^m A$;

c. $\mathcal{M}, w \models^m A \to B$ iff $\mathcal{M}, w \not\models^m A$ or $\mathcal{M}, w \models^m B$;

d. $\mathcal{M}, w \models^m \mathrm{K}A$ iff for every $v \in W$, $\mathcal{M}, v \models^m A$; (**ZP5**)
for every v such that wSv, $\mathcal{M}, v \models^m A$; (**ZP/ZCP**)

e. $\mathcal{M}, w \models^m \mathrm{B}A$ iff for every $v \in O$, $\mathcal{M}, v \models^m A$.

Validity and semantical consequence are defined as before, just with '$\models^m$' instead of plain '$\models$'. Now, in the following, let L be one of **ZP**, **ZP5** and **ZCP**.

Lemma L7. *If* Γ *is a saturated set, then, for some monoclustered model* $\mathcal{M}$, $\Gamma = [\mathcal{M},w]$.

Proof. Let us suppose that Γ is saturated. The best way to show the lemma is to take some subset of the set **S** of saturated sets which includes Γ as the set of worlds, and show that we can have a model on this—Γ would be then one of the worlds and we'd be done. Well, for **ZP** and **ZCP** we'll use again the relation μ we defined over the set **S** of all saturated sets, namely $\Sigma\mu\Delta$ iff $\varepsilon(\Sigma^K) \subseteq \Delta$. As one see from the proofs of P8.i and ii, this relation is reflexive and transitive.

[35] Trying to keep the S5-characteristics and extend ZP5 leads only to ZC5, which we already have thrown into (this story's) trash can.

(a) We first construct a model $\mathcal{M} = \langle [\Gamma]^\kappa, [\Gamma]^\beta [, \mu] \rangle$, where $[\Gamma]^\kappa = \{\Theta \in \mathbf{S} : \varepsilon(\Gamma^K) \subseteq \Theta\}$, $[\Gamma]^\beta = \{\Theta \in \mathbf{S} : \varepsilon(\Gamma^B) \subseteq \Theta\}$, and, for **ZP** and **ZCP**, μ is as above.

(b) We first prove (in **ZP5**) that, for every $\Sigma, \Theta \in [\Gamma]^\kappa$, $\varepsilon(\Sigma^K) = \varepsilon(\Theta^K)$. Let $A \in \varepsilon(\Sigma^K)$. By construction of $[\Gamma]^\kappa$, $\varepsilon(\Gamma^K) \subseteq \Sigma$. We then have $\mathrm{K}A \in \Sigma$; $\neg\mathrm{K}A \notin \Sigma$, $\mathrm{K}\neg\mathrm{K}A \notin \Gamma$, $\neg\mathrm{K}\neg\mathrm{K}A \in \Gamma$ and, since $\vdash \neg\mathrm{K}\neg\mathrm{K}A \to A$, $A \in \Gamma$. Hence $\varepsilon(\Sigma^K) \subseteq \Gamma$, from what it follows that $\varepsilon(\Sigma^K) = \varepsilon(\Gamma^K)$. By a similar reasoning, $\varepsilon(\Theta^K) = \varepsilon(\Gamma^K)$, from what it follows that $\varepsilon(\Sigma^K) = \varepsilon(\Theta^K)$.

(c) We now prove, still in **ZP5**, that, for every $\Sigma \in [\Gamma]^\kappa$, $\varepsilon(\Sigma^K) = \cap\{\Theta \in [\Gamma]^\kappa\}$. Well, $A \in \varepsilon(\Sigma^K)$ iff, from (b), for every $\Theta \in [\Gamma]^\kappa$, $A \in \varepsilon(\Theta^K)$ iff for every $\Theta \in [\Gamma]^\kappa$, $\mathrm{K}A \in \Theta$ iff for every $\Theta \in [\Gamma]^\kappa$, $A \in \Theta$ iff $A \in \cap\{\Theta \in [\Gamma]^\kappa\}$.

(d) We prove now, for **ZP** and **ZCP**, that for every $\Sigma \in [\Gamma]^\kappa$, $\varepsilon(\Sigma^K) = \cap\{\Theta \in [\Gamma]^\kappa : \Sigma\mu\Theta\}$.
It is clear that $\varepsilon(\Sigma^K) \subseteq \cap\{\Theta \in [\Gamma]^\kappa : \varepsilon(\Sigma^K) \subseteq \Theta\}$. On the other direction, let A be such that $A \notin \varepsilon(\Sigma^K)$; then $\mathrm{K}A \notin \Sigma$. Since Σ is a saturated set, $\Sigma \nvdash \mathrm{K}A$, so by P7 there is an A-saturated set Σ^* such that $\varepsilon(\Sigma^K) \subseteq \Sigma^*$. Then $A \notin \Sigma^*$. Now, since $\Sigma \in [\Gamma]^\kappa$, $\varepsilon(\Gamma^K) \subseteq \Sigma$; and, since μ is transitive, $\varepsilon(\Gamma^K) \subseteq \Sigma^*$; so $\Sigma^* \in \{\Theta \in \mathbf{S} : \varepsilon(\Gamma^K) \subseteq \Theta\}$, i.e., $\Sigma^* \in [\Gamma]^\kappa$. From this it follows that $A \notin \cap\{\Theta \in [\Gamma]^\kappa : \varepsilon(\Sigma^K) \subseteq \Theta\}$.

(e) We prove now that, for every $\Sigma \in [\Gamma]^\kappa$, $\varepsilon(\Sigma^B) = \cap\{\Theta \in [\Gamma]^\beta\}$. Let A be such that $A \in \varepsilon(\Sigma^B)$; then $\mathrm{B}A \in \Sigma$; $\neg\mathrm{B}A \notin \Sigma$. By the construction of $[\Gamma]^\kappa$, $\mathrm{K}\neg\mathrm{B}A \notin \Gamma$, $\neg\mathrm{B}A \notin \Gamma$, $\mathrm{B}A \in \Gamma$, and, finally, $A \in \varepsilon(\Gamma^B)$. Obviously then $A \in \cap\{\Theta \in [\Gamma]^\beta\}$ (by the construction of $[\Gamma]^\beta$), so $\varepsilon(\Sigma^B) \subseteq \cap\{\Theta \in [\Gamma]^\beta\}$.
On the other direction, let A be such that $A \notin \varepsilon(\Sigma^B)$; then $\mathrm{B}A \notin \Sigma$. By the construction of $[\Gamma]^\kappa$, $\mathrm{KB}A \notin \Gamma$, $\mathrm{B}A \notin \Gamma$, and $\Gamma \nvdash \mathrm{B}A$, so by P7 there is an A-saturated set Θ such that $\varepsilon(\Gamma^B) \subseteq \Theta$. Then $A \notin \Theta$ and, by the construction of $[\Gamma]^\beta$, $\Theta \in [\Gamma]^\beta$. Thus $A \notin \cap\{\Theta \in [\Gamma]^\beta\}$.
It follows that, for every $\Sigma \in [\Gamma]^\kappa$, $\varepsilon(\Sigma^B) = \cap\{\Theta \in [\Gamma]^\beta\}$.

(f) We prove now that $[\Gamma]^\beta \neq \emptyset$. Since Γ is a saturated set, there is some wff A for which Γ is A-saturated. So $\Gamma \nvdash A$, and obviously $\Gamma \nvdash \mathrm{K}A$, $\Gamma \nvdash \mathrm{BK}A$. By L1 there is some saturated set Σ such that $\varepsilon(\Gamma^B) \subseteq \Sigma$. So $\Sigma \in [\Gamma]^\beta$, and $[\Gamma]^\beta \neq \emptyset$. It follows immediately that also $[\Gamma]^\kappa \neq \emptyset$, because clearly $[\Gamma]^\beta \subseteq [\Gamma]^\kappa$.

(g) We now have to prove (in **ZP** and **ZCP**) that, if $\Theta \in [\Gamma]^\beta$ and $\Sigma \in [\Gamma]^\kappa$, then $\Sigma\mu\Theta$. Let A be such that $A \in \varepsilon(\Sigma^K)$; then $\mathrm{K}A \in \Sigma$; $\mathrm{B}A \in \Sigma$, and $A \in \varepsilon(\Sigma^B)$. By (f) above, $A \in \cap\{\Theta \in [\Gamma]^\beta\}$, so $A \in \Theta$ and $\varepsilon(\Sigma^K) \subseteq \Theta$. That is, $\Sigma\mu\Theta$.

(h) We now have to prove (in **ZCP**) that for every $\Sigma, \Theta \in [\Gamma]^\kappa$, if $\Sigma \in [\Gamma]^\beta$ and $\Sigma\mu\Theta$ then $\Theta \in [\Gamma]^\beta$. Since $\Sigma \in [\Gamma]^\beta$, $\varepsilon(\Gamma^B) \subseteq \Sigma$. Now, for every wff A such that $\mathrm{B}A \in \Gamma$, $\mathrm{BK}A \in \Gamma$, and $\mathrm{K}A \in \Sigma$. But then $A \in \Theta$, so $\varepsilon(\Gamma^B) \subseteq \Theta$ and $\Theta \in [\Gamma]^\beta$.

(i) We have now to prove that $\mathcal{M} = \langle [\Gamma]^\kappa, [\Gamma]^\beta [, \mu] \rangle$ is a monoclustered model. We already proved that $[\Gamma]^\kappa$ and $[\Gamma]^\beta$ are non-empty, and that μ satisfies the required conditions. We show now that $\mathcal{M}$ fulfills the conditions of definition D8. We say that $\mathcal{M}, \Sigma \models A$ iff $A \in \Sigma$, for every wff A and every $\Sigma \in [\Gamma]^\kappa$. Now we have, for all $\Sigma \in [\Gamma]^\kappa$,

(i) $\mathcal{M}, \Sigma \models \neg B$ iff $\neg B \in \Sigma$ iff $B \notin \Sigma$ iff $\mathcal{M}, \Sigma \nvDash B$;

(ii) $\mathcal{M}, \Sigma \models B \to C$ iff $B \to C \in \Sigma$ iff $B \notin \Sigma$ or $C \in \Sigma$ iff $\mathcal{M}, \Sigma \nvDash B$ or $\mathcal{M}, \Sigma \models C$;

(iii) $\mathcal{M}, \Sigma \models \mathrm{K}B$ iff $\mathrm{K}B \in \Sigma$ iff $B \in \varepsilon(\Sigma^K)$ iff

ZP5: $B \in \cap\{\Theta \in [\Gamma]^\kappa\}$ iff for all $\Theta \in [\Gamma]^\kappa$, $B \in \Theta$;

ZP/ZCP: $B \in \cap\{\Theta \in [\Gamma]^\kappa : \Sigma\mu\Theta\}$ for all $\Theta \in [\Gamma]^\kappa$ such that $\Sigma\mu\Theta$, $B \in \Theta$;

(iv) $\mathcal{M}, \Sigma \models \mathrm{B}B$ iff $\mathrm{B}B \in \Sigma$ iff $B \in \varepsilon(\Sigma^B)$ iff $B \in \cap\{\Theta \in [\Gamma]^\beta\}$ iff for all $\Theta \in [\Gamma]^\beta$, $B \in \Theta$.

Now obviously $\Gamma \in [\Gamma]^{K}$, hence Γ is one of the worlds in $\mathcal{M}$. Now we define $[\mathcal{M},\Gamma] = \{A\colon \mathcal{M},\Gamma \vDash A\}$, and obviously enough, $\Gamma = [\mathcal{M},\Gamma]$. ■

Theorem T16. $\vdash_L A$ *iff* $\vDash^m A$.

Proof.

(I) Suppose $\vdash_L A$.

(A) A is an axiom. We examine each case. If A is a tautology, it is evident from clauses D8b and D8c that $\vDash^m A$. And it is also obvious that (k), (t), (4) and (5), ali valid formulas, are also monoclustered valid. Let us now consider the belief case, and the mixed formulas.

(k^b) A is of the form $\mathrm{B}(p \to q) \to (\mathrm{B}p \to \mathrm{B}q)$. Let us suppose that A is not valid. Then there is a monoclustered model $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \mathrm{B}p$, $\mathcal{M}, w \vDash^m \mathrm{B}(p \to q)$ and $\mathcal{M},w \nvDash^m \mathrm{B}q$. It follows that there is $v \in O$ such that $\mathcal{M},v \nvDash^m q$. But it also follows that $\mathcal{M},v \vDash^m p$ and $\mathcal{M},v \vDash^m p \to q$, which is impossible. Thus A is valid.

(4^b) A is of the form $\mathrm{B}p \to \mathrm{BB}p$. If A is not valid then there is a model $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \mathrm{B}p$ and $\mathcal{M},w \nvDash^m \mathrm{BB}p$. Now it follows from D8e that there is $v \in O$ such that $\mathcal{M},v \nvDash^m \mathrm{B}p$. Again from D8e we have a $t \in O$, such that $\mathcal{M},t \nvDash^m p$. However, $\mathcal{M},w \vDash^m \mathrm{B}p$ implies that for every open world t, $\mathcal{M},t \vDash^m p$. Thus A is valid.

(5^b) A is of the form $\neg\mathrm{B}p \to \mathrm{B}\neg\mathrm{B}p$. If A is not valid, then there is a model $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \neg\mathrm{B}p$ and $\mathcal{M},w \nvDash^m \mathrm{B}\neg\mathrm{B}p$; thus $\mathcal{M},w \nvDash \mathrm{B}p$. From D8e it follows that there is a $v \in O$ such that $\mathcal{M},v \nvDash^m p$. From D8e again we have a $t \in W$ such that $\mathcal{M},t \nvDash^m \neg\mathrm{B}p$, hence $\mathcal{M},t \vDash^m \mathrm{B}p$. Now this entails that, for every open world, inclusive w, $\mathcal{M},w \vDash^m p$—a contradiction. Thus A is valid.

(d^b) A is of the form $\mathrm{B}p \to \neg\mathrm{B}\neg p$. If A is not valid, then there is a model $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \mathrm{B}p$ and $\mathcal{M},w \nvDash^m \neg\mathrm{B}\neg p$; thus $\mathcal{M},w \vDash^m \mathrm{B}\neg p$. Since O is not empty, there is at least a $v \in O$ such that $\mathcal{M},v \vDash^m p$. However, it follows from D8e that $\mathcal{M},v \vDash^m \neg p$, $\mathcal{M},v \nvDash p$, what cannot be. Thus A is valid.

(m) A is of the form $\mathrm{K}p \to \mathrm{B}p$. If A is not valid, then there is a model $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \mathrm{K}p$ and $\mathcal{M},w \nvDash^m \mathrm{B}p$. From D8e there is then a $v \in O$, such that $\mathcal{M},v \nvDash^m p$. However, $\mathcal{M},w \vDash^m \mathrm{K}p$ entails that p is true in every world: thus $\mathcal{M},v \vDash^m p$—a contradiction. Thus A is valid.

(p) A is of the form $\mathrm{B}p \to \mathrm{KB}p$. If A is not valid, then there is a $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \mathrm{B}p$ and $\mathcal{M},w \nvDash^m \mathrm{KB}p$. From D8d it follows that there is a $v \in W$ such that $\mathcal{M},v \nvDash^m \mathrm{B}p$. From D8e it follows now that there is a $t \in O$ such that $\mathcal{M},t \nvDash^m p$. But $\mathcal{M},w \vDash^m \mathrm{B}p$ entails that for every open world t, $\mathcal{M},t \vDash^m p$—a contradiction. Thus A is valid.

(c) A is of the form $\mathrm{B}p \to \mathrm{BK}p$. If A is not valid, then there is a $\mathcal{M} = <W,O[,S]>$ and $w \in W$, such that $\mathcal{M},w \vDash^m \mathrm{B}p$ and $\mathcal{M},w \nvDash^m \mathrm{BK}p$. From D8d it follows that there is a $v \in O$ such that $\mathcal{M},v \nvDash^m \mathrm{K}p$. From D7e and D8e it follows now that there is a $t \in O$ such that $\mathcal{M},t \nvDash^m p$. But $\mathcal{M},w \vDash^m \mathrm{B}p$ entails that for every open world t, $\mathcal{M},t \vDash^m p$—a contradiction. Thus A is valid.

(B) A was obtained by using MP from B and $B\rightarrow A$. Induction hypothesis: $\models^m B$ and $\models^m B\rightarrow A$. If there is a monoclustered model $\mathcal{M}$ such that $\mathcal{M}\not\models^m A$, there is a $w \in W$ such that $\mathcal{M},w \not\models A$. But $\mathcal{M},w \models^m B$, $\mathcal{M},w \models^m B\rightarrow A$, and this is contradictory. Thus, for all $\mathcal{M}$, $\mathcal{M}\models^m A$ and A is valid.

(C) $A = \mathrm{K}B$ was obtained from B using RK. Induction hypothesis: $\models^m B$. Now if some $\mathcal{M}\not\models^m A$, there is a $w \in W$ such that $\mathcal{M},w \not\models^m A$, i.e., $\mathcal{M},w \not\models^m \mathrm{K}B$. From D8d it follows then that there is a $t \in W$ such that $\mathcal{M},t \not\models^m B$—and this cannot obviously be the case. Thus, for all $\mathcal{M}$, $\mathcal{M}\models^m A$ and A is valid.

(II) Suppose that $\not\vdash_L A$. From P6 there is an A-saturated set Σ such that $\Gamma \subseteq \Sigma$. From L7 there is a monoclustered model $\mathcal{M}$ and w in W such that $\Sigma = \Delta_{\mathcal{M},w}$. Thus $\mathcal{M},w \not\models^m A$. It thus follows that $\not\models^m A$. ■

That, for example, (c) does not comes out valid in **ZP5** models we can see at once. Suppose $\mathrm{B}p\rightarrow \mathrm{BK}p$ is not valid, then there is a model $\mathcal{M} = \langle W,O\rangle$ and $w \in W$, such that $\mathcal{M},w \models \mathrm{B}p$ and $\mathcal{M},w \not\models \mathrm{BK}p$. From D8e it follows that there is a $v \in O$ such that $\mathcal{M},v \not\models \mathrm{K}p$. From D8d it follows now that there is a $t \in W$ such that $\mathcal{M}, t \not\models p$. Now we have from D8e and $\mathcal{M},t \models \mathrm{B}p$ that, for every open world s, $\mathcal{M}, s \models p$. But here we don't get a contradiction, because t is not necessarily open. By a similar reasoning we can show that things like $\mathrm{B}p\rightarrow p$ and $\mathrm{B}p\rightarrow \mathrm{K}p$ are also not valid.

Now let $\mathcal{M}$ be an L-monoclustered model, for some EDL-calculus $L \in \{\mathbf{ZP}, \mathbf{ZP5}, \mathbf{ZCP}\}$. We define the sets $K(\mathcal{M})$ and $B(\mathcal{M})$ of the known and believed facts in $\mathcal{M}$, respectively, as:

$$K(\mathcal{M}) = \{\, A\colon \mathcal{M}\models \mathrm{K}A \,\};$$
$$B(\mathcal{M}) = \{\, A\colon \mathcal{M}, w \models A \text{ for every } w \in O \,\}.$$

It is easy to see, from this definition, that:

(i) $A \in B(\mathcal{M})$ iff for all w in $\mathcal{M}$, $\mathcal{M},w \models \mathrm{B}A$ iff $\mathcal{M}\models \mathrm{B}A$;

(ii) $A \notin B(\mathcal{M})$ iff for all w in $\mathcal{M}$, $\mathcal{M},w \models \neg\mathrm{B}A$ iff $\mathcal{M}\models \neg\mathrm{B}A$.

For instance (ii): if $A \notin B(\mathcal{M})$ then there is a $w \in O$ such that $\mathcal{M}, w \not\models A$. If now there were a $t \in W$ such that $\mathcal{M}, t \models \mathrm{B}A$, then we would have that, for every $w \in O$, $\mathcal{M}, w \models A$, a contradiction. So for every $t \in W$, $\mathcal{M}, t \not\models \mathrm{B}A$; $\mathcal{M}, t \models \neg\mathrm{B}A$, and thus $\mathcal{M}\models \neg\mathrm{B}A$.

Proposition P14. *Let $\mathcal{M}$ be a monoclustered L-model. Then $B(\mathcal{M})$ is an L-stable set.*

Proof.
(st1) For all tautologies A, $\mathcal{M}\models A$; thus $\mathcal{M}\models \mathrm{B}A$, $A \in B(\mathcal{M})$. Let us now suppose that $A, A\rightarrow B \in B(\mathcal{M})$. I.e., $\mathcal{M}\models \mathrm{B}A$, $\mathcal{M}\models \mathrm{B}(A\rightarrow B)$. Using k^b, $\mathcal{M}\models \mathrm{B}B$, $B \in B(\mathcal{M})$. Thus $B(\mathcal{M})$ is closed under boolean operations.
(st2) $A \in B(\mathcal{M})$ iff $\mathcal{M}\models \mathrm{B}A$ iff $\mathcal{M}\models \mathrm{BB}A$ ($\vdash \mathrm{B}A \leftrightarrow \mathrm{BB}A$) iff $\mathrm{B}A \in B(\mathcal{M})$.
(st3) $A \notin B(\mathcal{M})$ iff $\mathcal{M}\not\models \mathrm{B}A$ iff $\mathcal{M}\models \neg\mathrm{B}A$ iff $\mathcal{M}\models \mathrm{B}\neg\mathrm{B}A$ ($\vdash \neg\mathrm{B}A \leftrightarrow \mathrm{B}\neg\mathrm{B}A$) iff $\neg\mathrm{B}A \in B(\mathcal{M})$.
(st4) Suppose $B(\mathcal{M})$ is inconsistent. Thus $A, \neg A \in B(\mathcal{M})$, thus $\mathcal{M}\models \mathrm{B}A$, $\mathcal{M}\models \mathrm{B}\neg A$. But $\mathcal{M}\models \mathrm{B}A$ implies that $\mathcal{M}\models \neg\mathrm{B}\neg A$ (using d^b), and this is a contradiction. Hence $B(\mathcal{M})$ is consistent. ■

Lemma L8. *Let $\mathcal{M}$ be a monoclustered model, and $w \in W$. Then there is a model $\mathcal{N}$ such that, for every formula A, $\mathcal{M},w \models A$ iff $\mathcal{N} \models A$.*

Proof. Let us have $[\mathcal{M},w]$ defined again as $\{A\text{: } \mathcal{M}, w \models A\}$. Now let $\mathcal{N}^* = \{\mathcal{N}\text{: } \mathcal{N} \models [\mathcal{M},w]\}$. We show that:

(i) $\mathcal{N}^* \neq \emptyset$. Suppose $\mathcal{N}^* = \emptyset$. Thus for all models $\mathcal{N}$, $\mathcal{N} \not\models [\mathcal{M},w]$. From this fact it follows that for all models $\mathcal{N}$, if $\mathcal{N} \models [\mathcal{M},w]$ then $\mathcal{N} \models \alpha \wedge \neg\alpha$. Thus $[\mathcal{M},w] \models \alpha \wedge \neg\alpha$, and $[\mathcal{M},w] \vdash \alpha \wedge \neg\alpha$, and it follows that $[\mathcal{M},w]$ is inconsistent, what cannot be. Hence $\mathcal{N}^* \neq \emptyset$.

(ii) Let us have $\mathcal{N}$, $\mathcal{N}' \in \mathcal{N}^*$. We show that $\mathcal{N} \approx \mathcal{N}'$. Suppose that $\mathcal{N} \not\approx \mathcal{N}'$. Then there is a wff B, $\mathcal{N} \models B$, $\mathcal{N}' \not\models B$, $\mathcal{N}' \models \neg B$. But it is easy to show (like in L3) that $[\mathcal{M},w]$ is a saturated set, thus it cannot be the case that, for both models, $\mathcal{N} \models [\mathcal{M},w]$ and $\mathcal{N}' \models [\mathcal{M},w]$. Hence $\mathcal{N} \approx \mathcal{N}'$.

It is clear, then, that for some $\mathcal{N} \in \mathcal{N}^*$ and for every formula A, $\mathcal{M},w \models A$ *iff* $\mathcal{N} \models A$. ■

Proposition P15. *Let S be an L-stable set. Then there is a monoclustered L-model $\mathcal{M}_S$, such that S = $B(\mathcal{M}_S)$.*

Proof. Let S be L-stable. Thus, for some $\Gamma \in$ S, S = $\varepsilon(\Gamma^B)$. From L7 there is $\mathcal{M}$ and $w \in W$, such that, for ali formulas A, $A \in \Gamma$ iff $\mathcal{M}$,w $\models A$. From L8 there is $\mathcal{M}_S$ such that $\mathcal{M}_S \models A$ iff $\mathcal{M}$,w $\models A$. We now show that S = B($\mathcal{M}_S$). So: $A \in$ S iff $A \in \varepsilon(\Gamma^B)$ iff B$A \in \Gamma$ iff $\mathcal{M}$,w $\models$ BA iff $\mathcal{M}_S \models$ BA iff $A \in B(\mathcal{M}_S)$.■

After having established which kind of relationship holds between monoclustered models and stable sets, we can now ask ourselves which is then the model in which Angela believes only α. As I mentioned, HM's elegant solution in terms of (S5-)Kripke models consists in just taking the union of all models in which α holds. In ZP5 here this is not so problematic, but what about the other cases, in which we have to cope with an accessibility relation S which is not an equivalence relation? We can easily construct two models such that the plain union of S_M and S_N is not, for instance, transitive. So what can we do?

A first way would be trying to define a stronger union operation, namely one in which additional pairs would be added to the union of the S relations, so that properties like transitivity and the like could be preserved. We can get this introducing the notion of a *closed union* of two models: if $\mathcal{M} = \langle M,O_M[,S_M]\rangle$ and $\mathcal{N} = \langle N,O_N[,S_N]\rangle$ are two L-monoclustered models, we say that the model $\mathcal{U} = \mathcal{M} \dot{\cup} \mathcal{N}$ is the *closed union* of $\mathcal{M}$ and $\mathcal{N}$ if $\mathcal{U}$ is a triple $\langle U,O_U[,S_U]\rangle$, where:

i. $U = M \cup N$;

ii. $O_U = O_M \cup O_N$;

iii. $S_U = \cap\, \{T \subseteq U \times U\text{: } S_M \cup S_N \cup \{\langle w,v\rangle\text{: } w \in U \text{ and } v \in O_U\} \subseteq T$ and such that T is reflexive, transitive and, for ZCP, it holds that if $w \in O_U$ and $\langle w,v\rangle \in T$ then $v \in O_U\}$.

Some words concerning this definition. The set U is the union of the universes of the two models; nothing new here. The idea behind the definition of S_U is that this relation should be the smallest subset of $U \times U$ containing $S_M \cup S_N$ that still fullfils the desired properties of the knowledge accessiblity relation. It must also contain the set $\{\langle w,v\rangle\text{: } w \in U \text{ and } v \in O_U\}$ (which takes care, in standard model terminology, of R being a subrelation of S). For ZP5, of course, the clause (iii) doesn't apply. We can easily prove that the intersection of all subsets of $U \times U$ respecting this condition is the smallest set. This ensures that the (closed) union of two models will still be a model. The set of open worlds, of course, is the union of the open worlds of the two models.

A second way of resolving the difficulty would be by means of defining a submodel relation. Let again $\mathcal{M} = <M, O_M[, S_M]>$ and $\mathcal{N} = <N, O_N[, S_N]>$ be two monoclustered models. We say that $\mathcal{M}$ is a *submodel* of $\mathcal{N}$ ($\mathcal{M} \leq \mathcal{N}$) if $M \subseteq N$, $O_M \subseteq O_N$, and $S_M \subseteq S_N$. (We also say that $\mathcal{N}$ is an *extension* of $\mathcal{M}$.)

Some properties of this submodel relation: first, '$\leq$' is clearly reflexive and transitive. '$\leq$' is also antisymmetric: if $\mathcal{M} \leq \mathcal{N}$ and $\mathcal{N} \leq \mathcal{M}$ holds, then as consequence $M = N$, $O_M = O_N$, and $S_M = S_N$. Hence $\mathcal{M}$ and $\mathcal{N}$ are the same model.

The next proposition shows how the truth of certain formulas is preserved under submodels or extensions.

Proposition P16. *Let $\mathcal{M} = <M, O_M[, S_M]>$ and $\mathcal{N} = <N, O_N[, S_N]>$ be two Kripke models and A a formula, such that $\mathcal{M} \leq \mathcal{N}$ and $dg(A) = 0$. Then:*

(a) *if* $\mathcal{N} \models KA$ *then* $\mathcal{M} \models KA$;

(b) *if* $\mathcal{N} \models BA$ *then* $\mathcal{M} \models BA$;

(c) *if* $\mathcal{M} \models \neg KA$ *then* $\mathcal{N} \models \neg KA$;

(d) *if* $\mathcal{M} \models \neg BA$ *then* $\mathcal{N} \models \neg BA$.

Proof. (a) Suppose $\mathcal{N} \models KA$ and $\mathcal{M} \not\models KA$; so there is a $w \in M$ such that $\mathcal{M}, w \not\models KA$. From this fact it follows that there is a $v \in M$ such that [wS_Mv and] $\mathcal{M}, v \not\models A$. However, we have that $M \subseteq N$ [and $S_M \subseteq S_N$], so $w, v \in N$ [and $<w,v> \in S_N$]. Since $dg(A) = 0$, $\mathcal{M}, v \models A$ iff $\mathcal{N}, v \models A$;[36] thus $\mathcal{N}, v \not\models A$, hence $\mathcal{N}, w \not\models KA$ and $\mathcal{N} \not\models KA$, against the hypothesis.

(b), (c) and (d) are provable in a similar way. ■

As we see, this proposition is provable exactly because $dg(A) = 0$. That the property doesn't need to hold if the formulas are modalized is shown in the next picture.

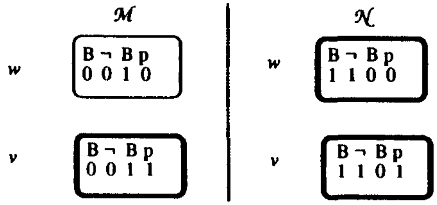

fig. 12

[36] This of course holds because A is a propositional formula, so its evaluation is independent from the values it may get in other worlds: we don't need to consider another world different from v.

Here we have $\mathcal{M} = <W,O_M [,S_M]>$ and $\mathcal{N} = <W,O_N [,S_N]>$. Open worlds are thicker outlined; thus $O_M = \{v\}$ and $O_N = \{w, v\}$. So $O_M \subset O_N$. (The S-relation doesn't matter.) The difference between the two models is that in $\mathcal{N}$ w is open, but not in $\mathcal{M}$. As a consequence of w also being open in $\mathcal{N}$ is that $\mathcal{N} \models \mathrm{B}\neg\mathrm{B}p$, but this is not the case in $\mathcal{M}$. The propositional variable p has in w and v the same value, and this, as the reader can see, doesn't hold anymore for the modalized formulas.

We can now use one of these two alternatives to characterize the state in which Angela believes only α. We can for instance take the set of ali monoclustered models in which $\mathrm{B}\alpha$ holds, and then prove that this set has a biggest element. It is not a surprise that this set is exaclty the closed union $\mathcal{M}_\alpha$ of all models. For a wff A, let $mds(A) = \{\mathcal{M} : \mathcal{M} \models A\}$.

Proposition P17. *If* $\mathcal{M}_\alpha = \textcircled{$\cup$}\ mds(\mathrm{B}\alpha)$, *then for each* $\mathcal{M} \in mds(\mathrm{B}\alpha)$, $\mathcal{M} \leq \mathcal{M}_\alpha$.

Proof. Let $\mathcal{M}_\alpha = \textcircled{$\cup$}\ mds(\mathrm{B}\alpha)$. Obviously for each $\mathcal{M} \in mds(\mathrm{B}\alpha)$, $W_M \subset W_{\mathcal{M}_\alpha}$, $O_M \subset O_{\mathcal{M}_\alpha}$, and $S_M \subset S_{\mathcal{M}_\alpha}$. Thus $\mathcal{M} \leq \mathcal{M}_\alpha$. Let us now suppose that there is $\mathcal{M}^+$ such that for each $\mathcal{M} \in mds(\mathrm{B}\alpha)$, $\mathcal{M} \leq \mathcal{M}^+$. But then $\mathcal{M}_\alpha \leq \mathcal{M}^+$, and $\mathcal{M}^+ \leq \mathcal{M}_\alpha$, with the consequence that $\mathcal{M}^+ = \mathcal{M}_\alpha$. ■

Thus Angela's belief state, when she believes only α, would be the set $B(\mathcal{M}_\alpha)$. As in [HM84], there are formulas that don't belong to $B(\mathcal{M}_\alpha)$, for instance our old acquaintance $\alpha = \mathrm{B}p \vee \mathrm{B}q$. Let us consider the ZP-models $\mathcal{M} = <M,O_M [,S_M]>$ and $\mathcal{N} = <N,O_N [,S_N]>$, where $\mathcal{M} = \{b,v\}$, $S_M = \{<b,b>, <b,v>, <v,b>, <v,v>\}$; $O_M = O_N = N = \{b,w\}$, $S_N = \{<b,b>, <b,w>, <w,b>, <w,w>\}$, and $b(p) = b(q) = 1$, $v(p) = 1$, $v(q) = 0$, $w(q) = 1$, $w(p) = 0$. Graphically, so that we can understand it better (the relation S is again not necessary):

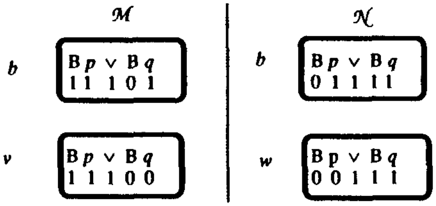

fig. 13

As we see, $\mathcal{M} \models \mathrm{B}\alpha$ and $\mathcal{N} \models \mathrm{B}\alpha$. Let now $\mathcal{M}_\alpha$ be the closed union of all models $\mathcal{M}^+$, such that $\mathcal{M}^+ \models \mathrm{B}\alpha$. Then we have $\mathcal{M}_\alpha \not\models \mathrm{B}\alpha$, because the set $\{b,v,w\}$ is contained in $O_{\mathcal{M}_\alpha}$ and thus there is a world, namely b, such that $\mathcal{M}_\alpha, b \not\models \mathrm{B}p$ (since $w \in O_{\mathcal{M}_\alpha}$, and $\mathcal{M}_\alpha, w \not\models p$) and $\mathcal{M}_\alpha, b \not\models \mathrm{B}q$ (since there is $v \in O_{\mathcal{M}_\alpha}$, and $\mathcal{M}_\alpha, v \not\models q$). Hence $\mathcal{M}_\alpha, b \not\models \mathrm{B}p \vee \mathrm{B}q$. Since $b \in O_{\mathcal{M}_\alpha}$, $\mathcal{M}_\alpha, b \not\models \mathrm{B}(\mathrm{B}p \vee \mathrm{B}q)$, that is, $\mathcal{M}_\alpha, b \not\models \mathrm{B}\alpha$, so $\mathcal{M}_\alpha \not\models \mathrm{B}\alpha$ and consequently $\alpha \notin B(\mathcal{M}_\alpha)$.

We can now introduce a second definition of honesty, based in monoclustered models: a formula α is L-$honest_M$ iff $\alpha \in B(\mathcal{M}_\alpha)$.

We must stress here that, like trying to find the minimal stable set, the method using monoclustered Kripke models doesn't have the advantage of working equally well for all EDL-systems, as one could expect. We don't have here, in fact, to introduce restrictions concerning the stable sets (like pr_K-, pr_B- and BT- sets), but, on the other hand, we didn't get monoclustered models for all logics.

2.4 An algorithmic approach

The third approach in trying to characterize knowledge states is done in HM through the use of an algorithm. In other words, the algorithm decides whether Angela knows a certain proposition B, given that she knows only α.

We try to do the same with the EDLs here. The idea is to generate, for each formula α, a set D^α which is the set of things Angela believes, if she believes only α. HM begin by asking themselves which formulas belong to D^α. In their case, since the knowledge logic is S5, any formula B for which $\mathrm{K}\alpha \rightarrow B$ holds must belong to D^α. Since, however, more than just the logical consequences of α should be in D^α, the algorithm ends up being the following:

$$B \in D^\alpha \qquad \text{iff} \qquad \models_{S5} \mathrm{K}\alpha \wedge \Psi_\alpha(B) \rightarrow B,$$

where $\Psi_\alpha(B)$ is the conjunction of $\mathrm{K}C$, for all subformulas $\mathrm{K}C$ of B for which $C \in D^\alpha$, and of $\neg\mathrm{K}C$, for all subformulas $\mathrm{K}C$ of B for which $C \notin D^\alpha$ (B being considered a subformula of itself). (cf. [HM84], p. 9)

The intuition behind the algorithm is that a formula B belongs to D^α iff it is a consequence of knowing α and the K-subformulas of B which have already been decided. So, for instance, a propositional formula C is in D^α iff $\models_{S5} \mathrm{K}\alpha \rightarrow C$.

After all is said and done, one could think that D^α is a stable set, but this doesn't always happen. Some of them would be inconsistent—the ones corresponding to dishonest formulas.

Well, how can all this apply to our EDL case here? The answer is: pretty much the same way, but changes are of course due to be made. First of all, obviously, if B is a consequence of believing α (i.e., $\mathrm{B}\alpha \rightarrow B$ holds), then surely B should be in D^α. However, taking the algorithm as it is would imply, for instance, that a propositional wff α, against our wishes, would not belong to D^α, because, obviously, $\mathrm{B}\alpha \rightarrow \alpha$ doesn't hold. But we can solve this by stating the following: if *believing* B is a consequence of believing α (i.e., $\mathrm{B}\alpha \rightarrow \mathrm{B}B$ holds), then B shall be in D^α. Of course, since $\mathrm{B}\alpha \rightarrow B$ entails that $\mathrm{B}\alpha \rightarrow \mathrm{B}B$, B will also be in D^α, if it is a consequence of believing α.

Besides, in the same way as in the knowledge case, not only logical consequences of believing only α will be in D^α. However, we cannot just take the "$\Psi_\alpha(B)$" part of the algorithm as it is, since we are working with logics that deal with knowledge and belief. So we should end up with the following:

$$B \in D^\alpha \qquad \text{iff} \qquad \models_L \mathrm{B}\alpha \wedge \Psi_\alpha(B) \wedge \Phi_\alpha(B) \rightarrow \mathrm{B}B,$$

where $\Psi_\alpha(B)$ is the conjunction of $\mathrm{K}C$ for all subformulas $\mathrm{K}C$ of B for which $C \in D^\alpha$, and $\neg\mathrm{K}C$ for all subformulas $\mathrm{K}C$ of B for which $C \notin D^\alpha$; $\Phi_\alpha(B)$ is the conjunction of $\mathrm{B}C$ for all subformulas $\mathrm{B}C$ of B for which $C \in D^\alpha$, and $\neg\mathrm{B}C$ for all subformulas $\mathrm{B}C$ of B for which $C \notin D^\alpha$; and where L is an EDL.

Same case as in HM, there are formulas α for which D^{α} is not consistent. For example (HM), $\alpha = p \wedge \neg Bp$. α is clearly consistent, but $B\alpha$, i.e., $B(p \wedge \neg Bp)$ implies both Bp and $\neg Bp$, so it is not consistent, and hence D^{α} is also inconsistent.

Moreover, even for a consistent α the set D^{α} might not be consistent. Again we consider our preferred example $\alpha = Bp \vee Bq$. It is easy to see that $\neg Bp \in D^{\alpha}, \neg Bq \in D^{\alpha}$, because it is not the case that $\models_L B\alpha \rightarrow Bp$, i.e., $\not\models_L B(Bp \vee Bq) \rightarrow Bp$, thus $\neg Bp \in D^{\alpha}$. For the same reason $\neg Bq \in D^{\alpha}$, and therefore $\neg Bp \wedge \neg Bq \in D^{\alpha}$. In view of this, we have that $B\alpha \wedge \Psi_{\alpha}(\alpha) \wedge \Phi_{\alpha}(\alpha) = B(Bp \vee Bq) \wedge \neg Bp \wedge \neg Bq$, and since $\models_L B(Bp \vee Bq) \wedge \neg Bp \wedge \neg Bq \rightarrow B(Bp \vee Bq)$, we get $\alpha \in D^{\alpha}$. Hence D^{α} is inconsistent.

This fact induces HM to give another definition of honesty based on the algorithm we have speaking of so far. We say that a formula α is *honest*$_D$ if the set D^{α} is L-consistent, for some EDL-system L. As we'll soon be proving, this new notion of honesty$_D$ is equivalent to the other two (for the logics to which they apply). We first prove the following proposition.

Proposition P18. *If α is honest$_D$ then D^{α} is a stable set.*

Proof. Let us suppose that α is honest$_D$. First, it is easy to see from the examples above, that (st2) and (st3) are satisfied, that is, $B \in D^{\alpha}$ iff $BB \in D^{\alpha}$ and $B \notin D^{\alpha}$ iff $\neg BB \in D^{\alpha}$. By the definition of honesty$_D$, D^{α} is consistent, so we have (st4). If then B is some propositional tautology, we have immediately that $\models_L B\alpha \rightarrow BB$, so $B \in D^{\alpha}$. Suppose now that $B \rightarrow C \in D^{\alpha}, B \in D^{\alpha}$, and $C \notin D^{\alpha}$. Since we have already proved that (st2) and (st3) hold, we have $B(B \rightarrow C) \in D^{\alpha}$, $BB \in D^{\alpha}$ and $\neg BC \in D^{\alpha}$, what implies that D^{α} is not consistent, and α would not be honest$_D$ against the hypothesis of the proposition. So (st1) also holds and we are done. ■

The algorithmic approach, then, seems to be the most promising of all, since it applies to all the logics considered here. Now I guess the reader is burning to state an objection—or at least a doubt. Remember stable sets, and how we didn't find a solution for some systems, say, Z? Why does it work here?

Well, the problem in trying to locate a minimum stable set via some set of propositional formulas had this drawback that, for instance in Z, there were many stable sets with this same propositional subset. Here we are not taking a lot of sets and trying to choose one—we are *building* a stable set from scratch. The way the algorithm works, it always chooses the path of most ignorance—if BA doesn't follow from the already decided formulas, then add $\neg$BA. So it is.

2.5 Putting it all together

After taking a look at all these different methods of characterizing minimal belief states, with of course different degrees of success, we can try to sum it all up and see what we get. The following table, in the first place, gives an overview of the different methods we have and which of our EDlogics they apply to.

logics	*stable sets*	*saturated sets*	*monoclustered models*	*algorithm*
Z	–	–	–	•
Z2	–	–	–	•
ZC	•	•	–	•
ZP	–	–	•	•
Z5	•	•	–	•
ZC2	•	•	–	•
ZCP	•	•	•	•
ZP5	•	•	•	•

We can now prove the equivalence of ali these definitions of honesty, what we do with the following theorem, which we also find in [HM84] (Theorem 2, p. 10). The proof is adapted from there. We prove the theorem only for the cases where $L \in \{$**ZP5**, **ZCP**$\}$, which are the only two logics in which all methods work. First we will need the following lemma:

Lemma L9.

(i) If $\mathcal{M} < \mathcal{N}$ then $B(\mathcal{N})$ is **ZP5**-*smaller than $B(\mathcal{M})$;*

(ii) If $\mathcal{M} \leq \mathcal{N}$ and $O_M \subset O_N$ then $B(\mathcal{N})$ is **ZCP**-*smaller than $B(\mathcal{M})$.*

Proof. Suppose $\mathcal{M} < \mathcal{N}$. That means $\mathcal{M} \leq \mathcal{N}$ but $\mathcal{M} \neq \mathcal{N}$. Let A be a zero-degree wff such that $A \in B(\mathcal{N})$. So $\mathcal{N} \models$ BA and, by P17, $\mathcal{M} \models$ BA, $A \in B(\mathcal{M})$. Thus $pr_B(B(\mathcal{N})) \subseteq pr_B(B(\mathcal{M}))$. Now let B be a zero-degree wff such that $B \in B(\mathcal{N})$ and K$B \in B(\mathcal{N})$. So $\mathcal{N} \models$ KB and, by P16, $\mathcal{M} \models$ KB, $\mathcal{M} \models$ BKB, $B \in B(\mathcal{M})$, K$B \in B(\mathcal{M})$. Thus $pr_K(B(\mathcal{N})) \subseteq pr_K(B(\mathcal{M}))$. Now let us consider the two logics separately:

(i) In **ZP5**, since $\mathcal{M} \leq \mathcal{N}$, we have that $M \subseteq N$ and $O_M \subseteq O_N$. Since $\mathcal{M} \neq \mathcal{N}$, we must have either $M \subset N$, or $O_M \subset O_N$. Suppose $O_M \subset O_N$. So there is a world w such that $w \notin O_M$ and $w \in O_N$. Now it is easy to show that there is some zero-degree formula A such that $\mathcal{N}, w \nvDash A$, but, for every $v \in O_M$, $\mathcal{M}, v \models A$. It follows that $\mathcal{M} \models$ BA, and $A \in B(\mathcal{M})$, but $\mathcal{N} \nvDash$ BA, $\mathcal{N} \models \neg$BA and $A \notin B(\mathcal{N})$. So $pr_B(B(\mathcal{N})) \subset pr_B(B(\mathcal{M}))$. Since we already have that $pr_K(B(\mathcal{N})) \subseteq pr_K(B(\mathcal{M}))$, $B(\mathcal{N})$ is **ZP5**-smaller than $B(\mathcal{M})$. Now suppose that $M \subset N$. So there is a world w such that $w \notin \mathcal{M}$ and $w \in \mathcal{N}$. If $w \in O_N$, then $O_M \subset O_N$, and the proof goes as before. So suppose $O_M = O_N$. We can easily show that there is some zero-degree formula A such that $\mathcal{N}, w \nvDash A$, but, for every $v \in M$, $\mathcal{M}, v \models A$. It follows that $\mathcal{M} \models$ KA, $\mathcal{M} \models$ BKA, and $A \in pr_K(B(\mathcal{M}))$. It also follows that $\mathcal{N} \nvDash$ KA, $\mathcal{N} \models \neg$KA, $\mathcal{N} \models$ B$\neg$KA and $\neg$K$A \in B(\mathcal{N})$, hence $A \notin pr_K(B(\mathcal{N}))$. So $pr_K(B(\mathcal{N})) \subset pr_K(B(\mathcal{M}))$. Since we already have that $pr_B(B(\mathcal{N})) \subseteq pr_B(B(\mathcal{M}))$, $B(\mathcal{N})$ is **ZP5**-smaller than $B(\mathcal{M})$.

(ii) In **ZCP**, since $\mathcal{M} \leq \mathcal{N}$, we have that $M \subseteq N$ and $O_M \subseteq O_N$. Since $\mathcal{M} \neq \mathcal{N}$, we must have either $M \subset N$, or $O_M \subset O_N$, or $S_M \subset S_N$. If now $O_M \subset O_N$, we can show as before that $pr_B(B(\mathcal{N})) \subset pr_B(B(\mathcal{M}))$, and that's enough to get that $B(\mathcal{N})$ is **ZCP**-smaller than $B(\mathcal{M})$.■

Theorem T17. [HM84] *A formula α is honest$_M$ iff it is honest$_D$ iff it is honest$_S$ iff it is honest$_K$.*

Proof. The proof is adapted from HM's one. We do a cycle of implications.

(a) honest$_M$ $\Rightarrow$ honest$_D$:

If α is honest$_M$ then $\mathcal{M}_\alpha$ is the maximum model that satisfies Bα. We need to show that $B(\mathcal{M}_\alpha) = D^\alpha$, what we do by proving, by induction on the structure of a formula B, that $B \in D^\alpha$ iff $B \in B(\mathcal{M}_\alpha)$. Let B be a propositional variable, and suppose $B \in D^\alpha$: then $\models_L$ B$\alpha \rightarrow$ BB (because $\Psi_\alpha(B)$ and $\Phi_\alpha(B)$ are obviously empty). Now it follows that $\mathcal{M}_\alpha \models_L$ B$\alpha \rightarrow$ BB and, since $\mathcal{M}_\alpha \models_L$ Bα, it follows that $\mathcal{M}_\alpha$ $\models_L$ BB and so $B \in B(\mathcal{M}_\alpha)$. So assume that, for proper subformula C of B, that $C \in D^\alpha$ iff $C \in B(\mathcal{M}_\alpha)$.

($\Rightarrow$) Suppose now that $B \in D^\alpha$. Then we have (by the definition of D^α) that $\models_L$ B$\alpha \wedge \Psi_\alpha(B) \wedge \Phi_\alpha(B) \rightarrow$ BB. Now for every conjunct of the form BC in $\Psi_\alpha(B)$, we must have by definition that $C \in D^\alpha$, and thus by the induction hypothesis, $C \in B(\mathcal{M}_\alpha)$, and hence $\mathcal{M}_\alpha, w \models_L$ BC, for every w in $\mathcal{M}$. In an analogous way, for every conjunct of the form $\neg$BC in $\Psi_\alpha(B)$, we must have $C \notin D^\alpha$, and thus by the induction hypothsis $C \notin B(\mathcal{M}_\alpha)$, and hence $\mathcal{M}_\alpha, w \models_L \neg$B$C$, for every w in $\mathcal{M}_\alpha$. So $\mathcal{M}_\alpha, w \models_L \Phi_\alpha(B)$. In the very same way we get that $\mathcal{M}_\alpha, w \models_L \Psi_\alpha(B)$. It follows that, for every w in $\mathcal{M}_\alpha$, $\mathcal{M}_\alpha, w \models_L$ B$\alpha \wedge \Psi_\alpha(B) \wedge \Phi_\alpha(B)$, and thus that $\mathcal{M}_\alpha, w \models_L$ BB. This also holds for every open world w, so $B \in B(\mathcal{M}_\alpha)$.

($\Leftarrow$) Suppose now that $B \in B(\mathcal{M}_\alpha)$, and that $B \notin D^\alpha$. Thus $\not\models_L$ B$\alpha \wedge \Psi_\alpha(B) \wedge \Phi_\alpha(B) \rightarrow$ BB, and hence $\not\models_L$ B$\alpha \wedge \Psi_\alpha(B) \wedge \Phi_\alpha(B) \rightarrow B$. We then must have some model $\mathcal{M} = \langle W, O\ [,S] \rangle$ such that $\mathcal{M} \models$ B$\alpha \wedge \Psi_\alpha(B) \wedge \Phi_\alpha(B)$, and $\mathcal{M} \not\models B$. So there is some w in $\mathcal{M}$ such that $\mathcal{M}, w \not\models B$; $\mathcal{M}, w \models \neg B$. Now obviously $\mathcal{M} \leq \mathcal{M}_\alpha$, so we have $W \subseteq W_\alpha$, and $w \in W_\alpha$. We now prove the following:

(†) for any proper subformula C of B, if $\mathcal{M}_\alpha, w \models \Psi_\alpha(C) \wedge \Phi_\alpha(C)$ and $\mathcal{M}, w \models \Psi_\alpha(C) \wedge \Phi_\alpha(C)$ then $\mathcal{M}_\alpha, w \models C$ iff $\mathcal{M}, w \models C$.

(i) C is a propositional variable, so $\Psi_\alpha(C)$ and $\Phi_\alpha(B)$ are obviously empty. Now, since worlds are assumed to be truth-value assignments to propositional variables, it is immediate that $\mathcal{M}_\alpha, w \models C$ iff $\mathcal{M}, w \models C$.

(ii) $C = \neg D$. By the induction hypothesis, $\mathcal{M}_\alpha, w \models D$ iff $\mathcal{M}, w \models D$; and obviously $\mathcal{M}_\alpha, w \models C$ iff $\mathcal{M}, w \models C$.

(iii) $C = D \rightarrow E$. By the induction hypothesis, $\mathcal{M}_\alpha, w \models D$ iff $\mathcal{M}, w \models D$; and $\mathcal{M}_\alpha, w \models E$ iff $\mathcal{M}, w \models E$. Obviously $\mathcal{M}_\alpha, w \models C$ iff $\mathcal{M}, w \models C$.

(iv) $C =$ KD. Then $\mathcal{M}_\alpha, w \models$ KD iff KD is one of the conjuncts of Ψ_α(KD) (since $\mathcal{M}_\alpha, w \vdash \Psi_\alpha$(K$D$) by hypothesis, and one of KD and $\neg$KD must be a conjunct of Ψ_α(KD)) iff $\mathcal{M}, w \models$ KD.

(v) $C =$ BD. Then $\mathcal{M}_\alpha, w \models$ BD iff KD is one of the conjuncts of Φ_α(BD) (since $\mathcal{M}_\alpha, w \vdash \Phi_\alpha$(B$D$) by hypothesis, and one of BD and $\neg$BD must be a conjunct of Φ_α(BD)) iff $\mathcal{M}, w \models$ BD.

It thus follows from (†) that $\mathcal{M}_\alpha, w \models \neg B$, against the hypothesis that $B \in B(\mathcal{M}_\alpha)$. Hence $B \in D^\alpha$.
Now, since $B(\mathcal{M}_\alpha) = D^\alpha$, D^α must be consistent, so α is honest$_D$.

(b) honest$_D$ $\Rightarrow$ honest$_S$:

If α is honest$_D$ then by P18 D^α is stable. By the construction of D^α, $\alpha \in D^\alpha$. Moreover, for any zero-degree formula B, we have that $B \in D^\alpha$ iff $\models_L$ B$\alpha \rightarrow$ BB. We must now show, for each logic L, that D^α is the L-smallest stable set containing α.

(i) In **ZCP**, this means that D^{α} must be the stable set containing α whose belief propositional subset is minimum. It is easy to see that, for every stable set S containing α, $pr_B(D^{\alpha}) \subset pr_B(\mathrm{S})$. For suppose there is a stable set S containing α and a propositional wff $A \notin pr_B(\mathrm{S})$ such that $A \in pr_B(D^{\alpha})$. Then, by construction of D^{α}, $\models_L \mathrm{B}\alpha \rightarrow \mathrm{B}A$. However, $\mathrm{B}\alpha \rightarrow \mathrm{B}A \in \mathrm{S}$ too, and, since stable sets are closed under boolean consequences, $\mathrm{B}A \in \mathrm{S}$, $A \in \mathrm{S}$. Suppose there is now a stable set T containing α such that $\mathrm{T} \neq D^{\alpha}$, but $pr_B(\mathrm{T}) = pr_B(D^{\alpha})$. In **ZCP** that cannot be the case, because (by T13) stable sets are uniquely determined by their propositional subsets.

(ii) In **ZP5**, we must show that, for every stable set S, either $pr_B(D^{\alpha}) \subset pr_B(\mathrm{S})$ and $pr_K(D^{\alpha}) \subseteq pr_K(\mathrm{S})$, or $pr_B(D^{\alpha}) = pr_B(\mathrm{S})$ and $pr_K(D^{\alpha}) \subset pr_K(\mathrm{S})$. In the same way as in the ZCP case, we prove that $pr_B(D^{\alpha}) \subseteq pr_B(\mathrm{S})$. Suppose now there is a propositional wff $A \notin pr_K(\mathrm{S})$ such that $A \in pr_K(D^{\alpha})$. So $\mathrm{K}A \in D^{\alpha}$. Then, by construction of D^{α}, $\models_L \mathrm{B}\alpha \rightarrow \mathrm{BK}A$. However, $\mathrm{B}\alpha \rightarrow \mathrm{BK}A \in \mathrm{S}$ too, and, since stable sets are closed under boolean consequences, $\mathrm{BK}A \in \mathrm{S}$, $\mathrm{K}A \in \mathrm{S}$, $A \in \mathrm{S}$, $A \in pr_K(\mathrm{S})$. Hence $pr_K(D^{\alpha}) \subseteq pr_K(\mathrm{S})$. If now $pr_B(D^{\alpha}) \neq pr_B(\mathrm{S})$, then D^{α} is automatically L-smaller than S. Suppose $pr_B(D^{\alpha}) = pr_B(\mathrm{S})$: we then have that $pr_K(D^{\alpha}) \subset pr_K(\mathrm{S})$, or else S and D^{α} are the same (by T14). So again D^{α} is L-smaller then S.
In both cases, we have that D^{α} is the L-smallest stable set containing α. Thus α is honest$_S$.

(c) honest$_S$ $\Rightarrow$ honest$_M$: Suppose that α is honest$_S$, but not honest$_M$. Since α is honest$_S$, there is an L-smallest stable set S such that $\alpha \in \mathrm{S}$. By P15 there is some model $\mathcal{M}_S$ such that $\mathrm{S} = B(\mathcal{M}_S)$.

We prove first in **ZP5** that $\mathcal{M}_S = \bar{\cup}\, \boldsymbol{mds}(\mathrm{B}\alpha)$. So suppose there is a model $\mathcal{N}$ of $\mathrm{B}\alpha$ such that $\mathcal{M}_S < \mathcal{N}$. By L9.i, $B(\mathcal{N})$ is ZP5-smaller than S, what cannot be. So, for every $\mathcal{N} \in \boldsymbol{mds}(\mathrm{B}\alpha)$, $\mathcal{N} \leq \mathcal{M}_S$. By P17, $\mathcal{M}_S$ is the closed union $\mathcal{M}_{\alpha}$ of ali models in which $\mathrm{B}\alpha$ holds. Now, since α is honest$_S$, $\alpha \in \mathrm{S}$, $\alpha \in B(\mathcal{M}_{\alpha})$, and hence α is honest$_M$.

Now in ZCP, let $\mathcal{M}^*$ be $\bar{\cup}\{\mathcal{M}: B(\mathcal{M}) = B(\mathcal{M}_S)\}$.We prove that $\mathcal{M}^* = \bar{\cup}\, \boldsymbol{mds}(\mathrm{B}\alpha)$. So suppose there is a model $\mathcal{N}$ of $\mathrm{B}\alpha$ such that $\mathcal{M}^* \leq \mathcal{N}$. If $O_{M^*} \subset O_N$ then by L9.ii we have that $B(\mathcal{N})$ is ZCP-smaller than S, what cannot be. So suppose that $O_{M^*} = O_N$. It is easy to show that, in this case, $B(\mathcal{N}) = B(\mathcal{M}^*)$, so $\mathcal{N} \in \{\mathcal{M}: B(\mathcal{M}) = B(\mathcal{M}_S)\}$. It follows that $\mathcal{N} \leq \mathcal{M}^*$, and $\mathcal{M}^* = \mathcal{N}$. Thus, for every $\mathcal{N} \in \boldsymbol{mds}(\mathrm{B}\alpha)$, $\mathcal{N} \leq \mathcal{M}^*$. By P17, $\mathcal{M}^*$ is the closed union $\mathcal{M}_{\alpha}$ of all models in which $\mathrm{B}\alpha$ holds. Now, since α is honest$_S$, $\alpha \in \mathrm{S}$, $\alpha \in B(\mathcal{M}_{\alpha})$, and hence α is honest$_M$.■

Intermezzo 1

With the end of Part I we have reached a considerable success concerning our main goal, which was to find a characterization method for minimal belief states. It's a pity it didn't happen in all cases with all methods, but, most interesting for us, there is an algorithm that we can use with all systems. Since one of my interests here are programming issues, we could now consider ways of getting the algorithm implemented. As we saw, the basis of it consists in having a decision procedure for the corresponding logic, so this is going to be our main concern in the first place. And since alethic and epistemic logics have a very similar structure—sometimes, as I already mentioned once or twice, they are the same, differences being found only in the way you interpret the operators—we could take a look at proof methods for modal logics as well.

In [Pel89] (Section 3, pp. 18ff) we find a discussion of several types of such proof methods. So we have, among the so-called *direct* methods, the tableau, resolution, and natural deduction methods, and, among the *indirect* ones, syntactic and semantic methods. We'll talk a little about tableau systems later in this work (particularly when implementing one), but what I would like primarily to investigate is the method of generalized truth-tables, which, I think, deserves a little more attention, even if, as we'll see, it is not so as efficient as other possible approaches.

So in the Part II of this dissertation we are going to take a look at valuation semantics and generalized truth-tables for several modal logics—as well as for an example EDL. After that we'll move on to some programming.

II

Valuation Semantics and Generalized Truth-Tables

3

Valuation semantics for normal modal logics

"That must be wonderful! I don't understand it at all."

3.1 An informal overview

The aim of this first section is to make an informal presentation of what is called *valuation semantics* for some systems of modal logic, and of its main byproduct, the *generalized truth-tables* (GTTs for short). I'd say this is a rather complicated kind of semantics—in comparison with possible-world semantics perhaps even an unintuitive one—so we'll begin take a look at its main ideas, how it is supposed to work, which are the differences relatively to possible-world semantics, and so on. We'll have afterwards a formal development of the whole.

I guess probably few people ever heard about valuation semantics, or still remember what it is, so I'd better tell what I know of the story. Valuation semantics were first introduced by Andréa Loparić, in a 1977 paper, for the modal propositional logic **K** (see [Lo77]). In order to give a brief description of what valuation semantics is, let us take as a starting point a semantics for the classical propositional logic **PL**: there we see that a model is nothing more than an assignment of truth-values to the propositional variables, since the value of complex formulas can be calculated if the value their subformulas have is known. We could also say, in other words, that a model for **PL** is a function f from wffs into truth-values obeying certain conditions (like $f(\neg A) \neq f(A)$, for instance).

If we now consider a possible-world semantics for some intensional logic, we notice that the structure of a model undergoes a deep change: one doesn't talk anymore about *only one* assignment (which, in a sense, describes a *possible world*), but about a whole set (a "universe") of them. The value of a formula whose main operator is an intensional one thus also depends on the value its subformulas get on various other worlds which are *accessible*. Here is where the famous *accessibility relations* come into the picture: formally, a model is now a triple $<W, R, V>$, where W denotes a set of worlds, R is a binary (acessibility) relation over W, and V is a function which takes arguments in formulas and worlds and goes into truth-values. The beauty of this construction is that one can get models for different modal logics by laying different conditions upon the relation R. (For instance, requiring of it to be *reflexive* singles out a class of models which characterizes the logic **T**.) On the other hand, in spite of models changing in this way, truth definitions for intensional operators like '$\Diamond$' (for "it is possible that...") are still given as usual, namely by means of necessary and sufficient conditions (*iff*-conditions: "$\Diamond A$ is true iff this-or-that holds").

Valuation semantics proceed the other way round: a model, which is called a *valuation*, is just one "world" (a function from wffs into {0,1} having some special properties); that is, one doesn't have to introduce a set of worlds and an accessibility relation. The change comes with respect to truth definitions for intensional operators, which now appear in the form "if $\Diamond A$ is true then such-and-such conditions hold; and if $\Diamond A$ is false then such-and-such other conditions hold".

One could argue, of course, about the propriety of the statement "a model is just one world", since, as it will be shown later, to evaluate a formula one also has to take other valuations (i.e.: other models) in consideration. More than that, when all is said and done a valuation ends up being proved to be the characteristic function of a maximal consistent set. In a sense, then, the whole could be like saying, in the setting of a possible-world semantics, that the only universe (model) you have to consider is the class of ali MCSs and, besides, you don't have to bother about introducing accessibility relations. This can be a question of seeing things this or that way. Later on we'll prove some kind of equivalence between valuation and possible-world semantics—which is not surprising at all, since the same formulas have to come out as valid. Well, if one asks my opinion, I would say the main difference lies on the fact that valuations are not declared *a priori* to be characteristic functions of MCSs; unlike possible-world models, they are defined inductively for certain sequences of formulas; it is only afterwards that they are generalized and proved to be characteristic functions of MCSs. And it is exactly because they are so defined that they generate in an easy way decision procedures, namely the GTTs, which allow us to examine *ali relevant models* to some formula.

Back to historical matters, Loparić and I gave, some years after her original paper, a valuation semantics for the minimal tense logic Kt ([LM84]; it was presented in 1980 as a short communication on the 4th Brazilian Conference on Mathematical Logic). In my master dissertation, under her supervision, I extended this semantics to several other tense logics as well, including here some naive logics combining time and modality. ([Mor82a, Mor82b]) In my dissertation there were also some problems left open, like to adequately define a valuation semantics for S4, still a tough and open case.[37]

But let us talk a little bit about GTTs. As we will see, one could argue about the propriety of the name "truth-table". They certainly neither are, nor pretend to be, *connective-defining truth-tables*—as we have, for instance, the one defining the truth-function "conjunction":

$\wedge$	1	0
1	1	0
0	0	0

fig. 14

We already know that intensional operators like "it is necessary that ..." are not truth-functional (where the value a formula gets depends exclusively on the values of its subformulas). Thus, if one takes the expression "truth-table" in this narrow sense, as meaning something that defines a truth-function, then GTTs are not truth-tables, but something else ("truth-tableaus", maybe). On the other hand, we also talk

[37] There is a "natural" definition of valuations for S4, but an important result couldn't be proved.

(perhaps by abuse of the language) about the truth-table for some formula A, like the following one for $a \to (b \to a)$:

a	b	$b \to a$	$a \to (b \to a)$
1	1	1	1
0	1	0	1
1	0	1	1
0	0	1	1

fig. 15

If we thus understand "truth-table" as denoting this kind of construction, then certainly GTTs deserve the name. As we will soon be seeing, with GTTs the procedure is pretty much the same as in the classical, truth-functional case: we also build, for some wff A, a sequence $A_1,\dots,A_n$ of its subformulas, where $A = A_n$ is the last element; next we assign values to the propositional variables, and after having done this we compute values for the remaining formulas of the sequence. The difference is that the value of a modalized A_i in a certain line j of the GTT now depends not only on the value in j of its subformulas, but also on the values which some other wffs can take in other lines. It should now not be surprising at all that through this construction one can also determine whether A is valid (meaning it is true on all lines) or not.

Well, one can discuss a lot about whether and in which way valuation semantics (with the corresponding GTTs) are something new, or whether they are just another way of presenting possible-world semantics, or semantic tableaus—whether they are, so to speak, possible-world semantics disguised in another clothes. Guess I'll better make my presentation, and let the reader judge by him- or herself. (We'll return briefly to this topic in chapter 6.)

3.2 Normal modal logics

I am going to present, in the remaining of this chapter, valuation semantics for some normal modal logics. The contents will be, first, resuming Andréa Loparić's original paper on the subject (for **K**, see [Lo77]), with small changes of my own, and second, also presenting some results I got in my master dissertation (for **KT**, **KTB**, **KT5**, see [Mor82a]), as well as, third, presenting some new, even if straightforward, extensions of these (**KD**, **KB**, **KDB**, **K45**, **KD45**).

I'll begin by introducing some notions that will be of general use here as well as in later chapters of this Part II. We'll still be considering a *propositional language*, which now we'll call L^m. It is like the language L of the first part, but now, instead of the epistemic operators 'K' and 'B', we have the alethic necessity '□'.

Wffs are defined in the usual way; and '**FOR**' still denotes the set of wffs. We introduce now the weak modal operator with the following definition:[38]

$Df\Diamond.\ \Diamond A \quad =_{df} \quad \neg\Box\neg A.$

Now an axiom basis for **PL** consist of the following axioms and rule of inference:

$A1.\quad A \rightarrow (B \rightarrow A)$
$A2.\quad (A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))$
$A3.\quad (\neg B \rightarrow \neg A) \rightarrow ((\neg B \rightarrow A) \rightarrow B)$
$MP.\quad A,\ A \rightarrow B\ /\ B$

A *normal modal logic* is then to be defined as an extension of **PL** which includes at least $Df\Diamond$, the following axiom schema:

$K.\ \Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$

and is closed under the following rule of inference:

$RN.\ \vdash A\ \ /\ \vdash \Box A.$[39]

Taking **K** as the minimal normal modal logic (i.e., the smallest extension of **PL** containing $Df\Diamond$, K and closed under *RN*), we can now build other systems by adding to it other axioms. In this chapter we are going to consider only logics which can be obtained by adding to **K** one or more of the following axiom schemas:

$D.\ \Box A \rightarrow \Diamond A$
$T.\ \Box A \rightarrow A$
$4.\ \Box A \rightarrow \Box\Box A$
$B.\ A \rightarrow \Box\Diamond A$
$5.\ \Diamond A \rightarrow \Box\Diamond A.$[40]

In general, we will have $\mathbf{KS_1...S_n}$ as the extension of **K** obtained with axiom schemas $S_1,...,S_n$ (in any order). For instance, **KTB** is **K** plus schemas T and B; **KT5** (or **K5T**) is **K** plus T plus 5. It can be proved, for instance, that **KT5** is the same as **KT4B**. Taking the equivalences in consideration, we arrive at the following picture (cf. [Lem77], p. 58, or [Ch80], p. 132) of 15 non-equivalent normal systems (an arrow means that the logic on the arrow's left is contained on the one on the right):

[38] Working with definitions makes proofs shorter and life in general easier. Now, even if I don't handle "it is possible that..." here as a primitive operator, I'll consider it so in the semantics part, to show how things can be done. By the way, in [Lo77], also in [Mor82], only necessity is considered.

[39] The reader wanting to know more about normal modal logics is kindly referred to [Ch80], a very readable book.

[40] With the exception of *B*, these axioms (in the epistemic-doxastic version) are already known from Part I where they have names like *d*, and d^b, for *D* here.

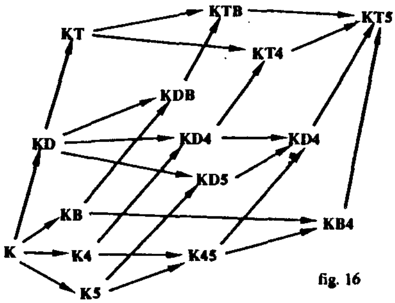

fig. 16

Some of these systems also have other names in the literature. Thus **KT**, **KT4**, **KTB**, **KT5**, **KD**, and **KD45** are also known as **T** (or **M**), **S4**, **B**, **S5**, **D**, and weak **S5**, respectively. In a possible-world semantics, models for these logics are obtained if we lay some constraints upon the accessibility relation *R*. For **K**, *R* could be any (binary) relation whatsoever, but for the other axioms the following conditions must hold of it:

T : reflexivity;
4 : transitivity;
B : symmetry;
5 : euclideanity;
D : seriality.

Definitions of *proof*, *theorem*, and *syntactical consequence*, for some normal modal logic *L*, are the same as in the epistemic-doxastic logical case (cf. Chapter 1), with the only care of substituting 'K' for '□', so I won't repeat them here. It is also woth mentioning that the Deduction Theorem (see T1) also holds here. Moreover, the following property—an analog of L1, with an (almost) identical proof—also hold for ali normal modal logics considered in this section:

Proposition P19. *If* $\Gamma \vdash A$ *then* $\Box\Gamma \cup \neg\Diamond\Gamma \vdash \Box A$ (*where* $\Box\Gamma = \{\Box B : B \in \Gamma\}$ *and* $\neg\Diamond\Gamma = \{\neg\Diamond B : \neg B \in \Gamma\}$).

As I said, valuations are going to be defined inductively over certain sequences of formulas, so we need first to characterize which sequences we are interested in. We say that a sequence $A_1,\ldots,A_n$ of formulas is a *normal sequence* of a logic *L* if, for $1 \leq i \leq n$, (a) if *B* is a subformula of A_i then there is $j < i$ such that $B = A_j$; and (b) for $1 \leq i \leq j$, if $A_i = A_j$ then $i = j$. Condition (a) ensures that, for every formula occurring in a sequence, ali its subformulas occur before it. Condition (b) ensures that we won't have unnecessary repetitions.[41]

[41] As one can see, normal sequences are just the plain, old sequences of formulas one learns in the school to construct if one is going to build a truth-table.

Now a valuation is supposed to be a function from the set FOR into the set $\{0,1\}$ of truth-values having certain properties and satisfying certain conditions—which conditions exactly will of course depend on the logic being considered. The basis of the whole construction are functions which satisfy the classical (extensional) conditions: so a function s is called a *semi-valuation* if s is a function from FOR into $\{0,1\}$ such that:

(a)	$s(\neg A) = 1$	iff	$s(A) = 0$;
(b)	$s(A \rightarrow B) = 1$	iff	$s(A) = 0$ or $s(B) = 1$.

It is now easy to prove that semi-valuations also have the following properties:

(c)	$s(A \wedge B) = 1$	iff	$s(A) = s(B) = 1$.
(d)	$s(A \vee B) = 1$	iff	$s(A) = 1$ or $s(B) = 1$.
(e)	$s(A \leftrightarrow B) = 1$	iff	$s(A) = s(B)$.

Thus a semi-valuation is, in fact, a model for the classical propositional logic **PL**. For the modal logics extending **PL** we need to add some clause or clauses which will take care of the modal operators. We'll do this in two steps: the first one is to define, for each logic L, the notion of *$A_1,...,A_n$–valuations for L*, where $A_1,...,A_n$ is a normal sequence. They form a subset of the set of semi-valuations, and are obtained inductively: we define first A_1–valuations, and then go on by laying upon the newly defined $A_1,...,A_i$–valuations some constraints each time we find a modalized formula. When the $A_1,...,A_n$–valuations are at last defined, we extend the construction to *all* normal sequences, thus getting the *valuations* for L. And that's it. Having defined valuations, one can go on doing business as usual: a formula is valid if it gets the value 1 in every valuation; the semantics can be proved correct and complete, and so on. As I'll show later, valuations happen to be the characteristic functions of MCSs, and one could have of course begun by defining them to be so, but doing things the way we do here gives us easily the GTTs and decidability.

3.3 Defining $A_1,...,A_n$–valuations

So the main point is to find, for each logic, a nice definition of an $A_1,...,A_n$–valuation that suits it. Before we do just this, I'll have to introduce some definitions and abbreviations which will be needed. In the following let us suppose that Γ is some set of formulas, and f a function from FOR into $\{0,1\}$. In a similar way to what we have done by the EDLs, we first define the sets of *necessities*, *possibilities* and *impossibilities* of Γ as:

$$\Gamma^{\Box} =_{df} \{A \in \Gamma: \text{ for some } B, A = \Box B\};$$

$$\Gamma^{\Diamond} =_{df} \{A \in \Gamma: \text{ for some } B, A = \Diamond B\};$$

$$\Gamma^{\neg\Diamond} =_{df} \{A \in \Gamma: \text{ for some } B, A = \neg\Diamond B\}.$$

As one see, they are the subsets of Γ containing wffs whose main operator is '$\Box$' or '$\Diamond$' or the combination '$\neg\Diamond$'. Next we define, for each of these sets, its *scope set*:

$$\begin{aligned}
\varepsilon(\Gamma^{\Box}) &=_{\mathrm{df}} \{A:\ \Box A \in \Gamma^{\Box}\};\\
\varepsilon(\Gamma^{\Diamond}) &=_{\mathrm{df}} \{A:\ \Diamond A \in \Gamma^{\Diamond}\};\\
\varepsilon(\Gamma^{\neg\Diamond}) &=_{\mathrm{df}} \{\neg A:\ \neg\Diamond A \in \Gamma^{\neg\Diamond}\}.
\end{aligned}$$

Now we define what it means for a function f to *satisfy* (*reject*) a set of formulas:

$$f \vDash_u \Gamma \quad =_{\mathrm{df}} \quad \text{for every } A \in \Gamma, \text{ and for } u \in \{0,1\}, f(A) = u.$$

Of course, it only seems to be correct to speak of satisfiability—like "f satisfies Γ"—in the case of $\vDash_1$ (that is, if $u = 1$). So in the case of $\vDash_0$ I decided to say that "f rejects Γ". Just note, however, that "not satisfying" (when f gives 0 to at least one of the wffs in Γ) doesn't mean the same as "rejecting" (when f gives 0 to all of them).

Next we define the subset of Γ having value u according to f as:

$$\Gamma_{f,u} \quad =_{\mathrm{df}} \quad \{A \in \Gamma:\ f(A) = u\}.$$

And at last some abbreviations. First of all, it's going to be quite a job for me having to type—and for you having to read—things like '$A_1,\ldots,A_n$' every second line. So let us agree on the following convention (**Abb1**): we will use 'α' as a typographical substitution for '$A_1,\ldots,A$', so, when we write 'α_{k-1}' and 'α_n', what we mean is actually '$A_1,\ldots,A_{k-1}$' and '$A_1,\ldots,A_n$', and so forth.

In the second place we have the other abbreviation, which will be meaning different things for the different logics, so please pay attention. **Abb2**: Let α_n be a normal sequence (i.e.: $A_1,\ldots,A_n$) and f, g two functions from FOR into $\{0,1\}$. We say that, for $1 \leq k \leq n$,

(a) for **K**, **KD**, **KT**:

$f<k>g$ iff $g \vDash_1 \varepsilon(\{\alpha_k\}^{\Box}_{f,1})$, and
$g \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}_{f,0})$;

(b) for **KB**, **KDB**, **KTB**:

$f<k>g$ iff $g \vDash_1 \varepsilon(\{\alpha_k\}^{\Box}_{f,1})$,
$g \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}_{f,0})$,
$f \vDash_1 \varepsilon(\{\alpha_k\}^{\Box}_{g,1})$, and
$f \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}_{g,0})$;

(c) for **K45**, **KD45**:

$f<k>g$ iff $g \vDash_1 \varepsilon(\{\alpha_k\}^{\Box}_{f,1})$,
$g \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}_{f,0})$,
$\{\alpha_k\}^{\Box}_{f,1} = \{\alpha_k\}^{\Box}_{g,1}$, and
$\{\alpha_k\}^{\Diamond}_{f,0} = \{\alpha_k\}^{\Diamond}_{g,0}$;

(d) for **KT5**:

$f<k>g$ iff $\{\alpha_k\}^{\Box}_{f,1} = \{\alpha_k\}^{\Box}_{g,1}$, and
$\{\alpha_k\}^{\Diamond}_{f,0} = \{\alpha_k\}^{\Diamond}_{g,0}$.

Some words about this all. First, this abbreviation (showing only the "k" parameter) can be used without fear of confusion if we are working with a fixed normal sequence. If needed, we can also write $f<\alpha_k>g$—or even $f<A_1,...,A_k>g$, which is more precise. Now to understand what exactly is at stake here, let us consider the sets involved, beginning with (a). '$\{\alpha_k]^{\Box}$' denotes the subset of α_k consisting of those wffs in this sequence whose main operator is a necessary; the subscript '$_{f,1}$' now forms a new set by choosing those wffs among them which are given 1 by f. We take now the scope of this set, and to this formulas the function g must give a 1, if $f<k>g$ is to obtain. Similarly for the second half of (a), only that we are now dealing with impossibilities which must be rejected. For the other cases, things are pretty much the same: variations on a theme.

One can think, if one wishes, of '$f<k>g$' as representing a kind of "accessibility relation" between two functions, what is not entirely wrong, just a little bit. The idea is that, when $f<k>g$ obtains, for instance in (a), g satisfies the scope set of f's true necessities, and rejects the scope set of its impossibilities. This is similar to what happens on possible-world models if we have two worlds x and y such that xRy. The difference is, we don't have here an accessibility relation *simpliciter*, since $f<k>g$ holds between this two functions just for the small set of formulas being part of the normal sequence $A_1,...,A_k$ (which is all we need to evaluate some wff): this relation may not hold anymore if we consider a longer sequence $A_1,...A_k,A_{k+1},...,A_{k+j}$. Moreover, contrary to the possible-world semantics, we don't introduce worlds and relations as primitive elements of models, so we don't have to bother about what worlds "really are" and what accessibility "really means". And yet another remark: in (a), (b) and (c), for instance, the $f<k>g$ abbreviation is the same for two or three different logics. Thinking of $f<k>g$ as being an accessibility relation would imply that it should mean different things, or have different properties, for each particular logic, what is not the case here. In fact, only the way in which we compute the value of modalized formulas will allow us to make the differences (see below) for logics in which "$f<k>g$" means the same thing.

But let us now proceed and take a look (at last!) to our main definition. I'll give first the definitions for **K**, and then we'll see which changes are needed for the other logics.

Definition D9. v is a α_n*–valuation* (for **K**) if α_n is a normal sequence and:

1) $n = 1$ and v is a semi-valuation;

2) $n > 1$, v is an α_{n-1}–valuation and, if for some $m < n$,

A) $A_n = \Box A_m$,

I) if $v(A_n) = 0$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v<n\text{-}1>v_n$;

II) if $v(A_n) = 1$ then for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$] there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$ [$v_p(A_q) = 1$], $v_p(A_m) = 1$ and $v<n\text{-}1>v_p$.

B) $A_n = \Diamond A_m$,

I) if $v(A_n) = 1$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 1$ and $v<n\text{-}1>v_n$;

II) if $v(A_n) = 0$ then for every p, every q, $q < p \leq n$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$ [$A_p = \Box A_q$ and $v(A_p) = 0$] there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$ [$v_p(A_q) = 0$], $v_p(A_m) = 0$ and $v<n\text{-}1>v_p$.

This definition certainly looks scaring, so let us go slowly through it. Clause 1) gives the basis for the inductive definition: we have there a normal sequence with just one element (which must be by definition a propositional variable; else there would have been some subformula of it occurring before), and so everything required of v is that it shall be a semi-valuation. By clause 2), for $n > 1$, if the main operator of A_n is not a modal one, nothing has to be done, because the semi-valuation properties already take care of the extensional operators. And so we come to the case where $A_n = \Box A_m$. If v gives 0 to it, then we have to look for another α_{n-1}–valuation v_n giving 0 to A_m and satisfying/rejecting the scope sets of v. One can draw here a parallel to possible-world semantics, where there must be an accessible world falsifying A_m. If, on the other hand, v gives 1 to $\Box A_m$, we only require of v that it has had a "good behavior" before, i.e., that for every false necessity, or true possibility, the condition corresponding to case I) was satisfied. Here there is a difference in relation to possible-world semantics, where we require, for a true necessity $\Box A$, that its scope A gets truth in all accessible worlds. With valuation semantics this is not the case: we can have $v(\Box A_m) = 1$ and, nevertheless, it may exist some "accessible" α_{n-1}–valuation v_n with $v_n(A_m) = 0$ and $v{<}n\text{-}1{>}v_n$. Our requirements are thus weaker.

The case where $A_n = \Diamond A_m$ is similar, only with reversed values. Of course, since possibility is not a primitive operator, we might have not considered it here, what would have made the definition shorter.

Having thus defined α_n–valuations, the rest follows in a more or less straigthforward way. We can now say that a function v from FOR into $\{0,1\}$ is a *valuation* iff for every normal sequence α_n, v is an α_n–valuation. The next steps would now consist in getting some results about valuations, and then taking a look on how to prove correctness and completeness. But I would like first to show which kind of modifications are needed in order to get valuations for the other normal modal logics as well.

Actually the changes are not that big. I am not going to repeat the whole definition; just the places where changes are needed. It goes as follows:

- for **KB, K45:**

— *same as* **K**.

- for **KD, KDB, KD45:**

A) $A_n = \Box A_m$,

I) (*as in* **K**);

II) if $v(A_n) = 1$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 1$ and $v{<}n\text{-}1{>}v_n$; moreover, for every p, every q, ... (*as in* **K**);

B) $A_n = \Diamond A_m$,

I) (*as in* **K**);

II) if $v(A_n) = 0$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v{<}n\text{-}1{>}v_n$; moreover, for every p, every q, ... (*as in* **K**).

- for **KT, KTB, KT5:**

A) $A_n = \Box A_m$,

I) (*as in* **K**);

II) if $v(A_n) = 1$ then $v(A_m) = 1$ and for every p, every q, ... (*as in* **K**);

B) $A_n = \Diamond A_m$,

I) (*as in* **K**);

II) if $v(A_n) = 0$ then $v(A_m) = 0$ and for every p, every q, ... (*as in* **K**).

Now the definition for **K**, **KB** and **K45** is only superficially the same: remember, '$f<k>g$' abbreviates in each of these logics something different! The same holds for the **KD**, **KDB**, **KD45**, and for the **KT**, **KTB** and **KT5** definitions.

Well, what we did until now was to define an α_n–valuation for a normal sequence α_n, but sure we would like to consider longer sequences and so be able to extend this construction to an α_{n+1}–valuation, an α_{n+2}–valuation, and so forth. There are ways of doing this, but useful is going to be a particular kind of extension which will be called a *canonical extension*, and whose definition is the same for all our logics:

Definition D10. Let α_n be a normal sequence and v an α_{n-1}–valuation. We say that v_c is the *canonical extension* of v to α_n if:

A) for all $m < n$, $A_n \neq \Box A_m$, $A_n \neq \Diamond A_m$ and $v_c = v$; or

B) for some $m < n$, $A_n = \Box A_m$ $[A_n = \Diamond A_m]$ and v_c is a function from **FOR** into $\{0,1\}$ such that, for every formula B,

1) if A_n is not a subformula of B, then $v_c(B) = v(B)$;

2) if A_n is a subformula of B, then

a) for $B = A_n$, $v_c(B) = 0$ $[v_c(B) = 1]$ iff there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$ $[v^+(A_m) = 1]$ and $v<n\text{-}1>v^+$;

b) for $B = \neg C$, $v_c(B) = 1$ iff $v_c(C) = 0$;

c) for $B = C \rightarrow D$, $v_c(B) = 1$ iff $v_c(C) = 0$ or $v_c(D) = 1$;

d) for $B = \Box C$ or $B = \Diamond C$, $v_c(B) = v(B)$.

We have now to show that canonical extensions satisfy the requirements of Definition D9, i.e., that they are α_n–valuations too. In this we will use the notion of *normality*. Let v be an α_n–valuation: for $1 \leq k \leq n$, we say that v is $\Box_0$–α_k–*normal* if for every p, every q, $q < p \leq k$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is an α_k–valuation v_p such that $v_p(A_q) = 0$ and $v<k>v_p$. We say that v is $\Diamond_1$–α_k–*normal* if for every p, every q, $q < p \leq k$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, there is an α_k–valuation v_p such that $v_p(A_q) = 1$ and $v<k>v_p$.

This definition of normality applies not only to **K**, but also to every normal modal logic here considered, and it corresponds to the condition required on clause I) of the defintion of an α_n–valuation. However, for systems other than **K**, **KB** and **K45**, we also need other kinds of normality, namely those corresponding to the special conditions occuring in clause II). So we have, for **KD**, **KDB** and **KD45**, that a valuation v is $\Box_1$–α_k–*normal* if for every p, every q, $q < p \leq k$, such that $A_p = \Box A_q$ and $v(A_p) = 1$ there is an α_k–valuation v_p such that $v_p(A_q) = 1$ and $v<k>v_p$; v is $\Diamond_0$–α_k–*normal* if for every p, every q, $q < p \leq k$, such that $A_p = \Diamond A_q$ and $v(A_p) = 0$ there is an α_k–valuation v_p such that $v_p(A_q) = 0$ and $v<k>v_p$. As the reader has probably grasped by now, this condition is the one required to render axiom schema D valid.

In the case of **KT**, **KTB** and **KT5**, we have: v is $\Box_1$–α_k–*normal* if for every p, every q, $q < p \le k$, such that $A_p = \Box A_q$ and $v(A_p) = 1$, $v(A_q) = 1$; v is $\Diamond_0$–α_k–*normal* if for every p, every q, $q < p \le k$, such that $A_p = \Diamond A_q$ and $v(A_p) = 0$, $v(A_q) = 0$. (This is the condition which takes care of axiom schema *T*.)

Thus one can see that, even if some logics use '$f{<}k{>}g$' to abbreviate the same property (as we had for **K**, **KD**, and **KT**), the $\Box_1$– and $\Diamond_0$–normality requirements are different for each of them. In the following table I try to give an overall view of all these differences:

Logic	$f{<}k{>}g$	clause II
K	(a)	–
KD	(a)	w
KT	(a)	s
KB	(b)	–
KDB	(b)	w
KTB	(b)	s
K45	(c)	–
KD45	(c)	w
KT5	(d)	s

fig. 17

Some explanations. The letters (a), (b), (c), and (d) in the field '$f{<}k{>}g$' refers to the meaning of the corresponding abbreviations; that is, cases (a) through (d) of Abb2. "Clause II" (of the definition of an α_n–valuation) shows what kind of $\Box_1$– and $\Diamond_0$–normality are required in each logic: namely none ("—"), KD–type ("weak") or KT–type ("strong").

Now we are ready to get some results.

Lemma L10. *If v is an α_{n-1}–valuation and v_c is the canonical extension from v to α_n, then v_c is a semi-valuation.*

Proof. Straightforward: just consider that, for $i < n$, $v_c(A_i) = v(A_i)$, and v is a semi-valuation. For $i < n$, clauses b) and c) of D10 ensure that the classical properties are respected. So v_c is a semi-valuation. ■

Proposition P20. *Let α_n be a normal sequence, v an α_{n-1}–valuation and v_c the canonical extension of v to α_n. Let us suppose, for all systems, that v is $\Box_0$– and $\Diamond_1$–α_{n-1}–normal, and, for* **KD**, **KDB**, **KD45**, **KT**, **KTB** *and* **KT5**, *that v is $\Box_1$– and $\Diamond_0$–α_{n-1}–normal. In this case, v_c is an α_n–valuation.*

Proof. First of all, v_c is an α_{n-1}–valuation, because it is a semi–valuation and, by construction, for $1 \le i < n$, $v_c(A_i) = v(A_i)$. Now, if, for every $m < n$, $A_n \ne \Box A_m$, $A_n \ne \Diamond A_m$, v_c fulfills every condition of Definition D9, so it is an α_n–valuation. Suppose, then, that for some $m < n$, $A_n = \Box A_m$. We have two cases:

(I) $v_c(A_n) = 0$. By D10.B.2.a there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$ and $v{<}n\text{-}1{>}v^+$. Since v and v_c agree for $i < n$, $v_c{<}n\text{-}1{>}v^+$. So v_c is an α_n–valuation.

(II) $v_c(A_n) = 1$. **We consider separately the different systems:**

α) K, KB, K45:

(†) By D10.B.2.a, for every $\alpha_{n\text{-}1}$–valuation v^+ such that $v{<}n\text{-}1{>}v^+$, $v^+(A_m) = 1$.

Suppose now there is $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$ [or $A_p = \Diamond A_q$ and $v_c(A_p) = 1$]. Then $v(A_p) = 0$ [$v(A_p) = 1$] and, since v is $\Box_0$– and $\Diamond_1$–$\alpha_{n\text{-}1}$–normal, there is an $\alpha_{n\text{-}1}$–valuation v_p such that $v{<}n\text{-}1{>}v_p$ and $v_p(A_q) = 0$ [$v_p(A_q) = 1$]. Since v and v_c agree for $i < n$, we have that $v_c{<}n\text{-}1{>}v_p$. Now, from (†), we have that $v_p(A_m) = 1$ (eise we would have $v_c(A_n) = 0$). It follows, in this case, that v_c is an α_n–valuation.

β) KD, KDB, KD45:

If there is $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$ [or $A_p = \Diamond A_q$ and $v_c(A_p) = 1$], we prove as in α) that the conditions are fulfilled. We have now to prove that there is an $\alpha_{n\text{-}1}$–valuation v_n such that $v_n(A_m) =$ I and $v{<}n\text{-}1{>}v_n$. If there is some $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$, or $A_p = \Diamond A_q$ and $v_c(A_p) =$ I, then we have already proved it: there is an $\alpha_{n\text{-}1}$–valuation v_p such that $v{<}n\text{-}1{>}v_p$ and $v_p(A_q) = 0$ [$v_p(A_q)$ = I] and $v_p(A_m) = 1$ Suppose then that there is no $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$, or $A_p = \Diamond A_q$ and $v_c(A_p) = 1$. We have two possibilities:

i) there is some $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 1$, or $A_p = \Diamond A_q$ and $v_c(A_p) = 0$. Then $v(A_p) = 1$ (or 0) and, since v is $\Box_1$– and $\Diamond_0$–$\alpha_{n\text{-}1}$–normal, there is an $\alpha_{n\text{-}1}$–valuation v_p such that $v{<}n\text{-}1{>}v_p$ and $v_p(A_q) = 1$ (or 0). Since v and v_c agree for $i < n$, we have that $v{<}n\text{-}1{>}v_p$; and it follows from (†) that $v_p(A_m) = 1$.

ii) there is no $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 1$, or $A_p = \Diamond A_q$ and $v_c(A_p) = 0$. Well, in this case, $\{\alpha_k\}^{\Box} = \{\alpha_k\}^{\Diamond} = \emptyset$, in which case $v_c \vDash_1 \varepsilon(\{\alpha_k\}^{\Box}{}_{v_c,1})$ and $v_c \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v_c,0})$; so $v_c{<}n\text{-}1{>}v_c$ and, from (†), $v_c(\mathrm{A_m}) = 1$. It follows, in this case, that v_c is an α_n–valuation.

γ) KT, KTB, KT5:

If there is $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$ [or $A_p = \Diamond A_q$ and $v_c(A_p) = 1$], we prove as in α) that the conditions are fulfilled. We have now to prove that $v_c(A_m) = 1$. Since, for every $\alpha_{n\text{-}1}$–valuation v^+ such that $v{<}n\text{-}1{>}v^+$, $v^+(A_m) = 1$, we only need to prove that $v_c{<}n\text{-}1{>}v_c$. In KT5 this is immediate, because $\{\alpha_k\}^{\Box}{}_{v_c,1} = \{\alpha_k\}^{\Box}{}_{v_c,1}$ and $\{\alpha_k\}^{\Diamond}{}_{v_c,0} = \{\alpha_k\}^{\Diamond}{}_{v_c,0}$. For KT and KTB, we make use of the fact that v is $\Box_1$– and $\Diamond_0$–$\alpha_{n\text{-}1}$–normal. For every $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 1$ [$A_p = \Diamond A_q$ and $v_c(A_p) = 0$], we have that $v(A_p) = 1$ [$v(A_p) = 0$], and it follows from v's normality that $v(A_q) = 1$ [$v(A_q) = 0$]. So $v \vDash_1 \varepsilon(\{\alpha_k\}^{\Box}{}_{v,1})$ and $v \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0})$; it follows that $v{<}n\text{-}1{>}v$, and, since v and v_c agree for $i < n$, $v_c{<}n\text{-}1{>}v_c$ and we are done. Hence v_c is an α_n–valuation.

If now, for some $m < n$, $A_n = \Diamond A_m$, the proof goes in a similar way. ■

We have thus proved that canonical extensions are α_n–valuations under the assumption that the $\alpha_{n\text{-}1}$–valuations they are extending are normal. With the next lemma, we can show that α_n–valuations are normal without restrictions, and thus that they can be extended as long we we want them to be. Just remember that $\Box_1$– and $\Diamond_0$–normality doesn't apply to K, KB and K45, only to the other systems.

Lemma L11. (*Normality Lemma*) *Let v be an α_n–valuation. Then v is $\Box_0$–, $\Box_1$–, $\Diamond_0$– and $\Diamond_1$–α_n–normal.*

Proof. By induction on n. For $n = 1$ it holds trivially, so let $n > 1$ and let us suppose that every α_{n-1}–valuation is $\Box_0$–, $\Box_1$–, $\Diamond_0$– and $\Diamond_1$–α_{n-1}–normal. It follows then from P20 that

(†) The canonical extensions of α_{n-1}–valuations to α_n are α_n–valuations.

We have now three cases:

(1) For every $m < n$, $A_n \neq \Box A_m$, $A_n \neq \Diamond A_m$. So v is trivially $\Box_0$–, $\Box_1$–, $\Diamond_0$– and $\Diamond_1$–α_n–normal.

(2) Let us suppose that, for some $m < n$, $A_n = \Box A_m$.

(I) Let $v(A_n) = 0$. We have:

1) $\{\alpha_n\}^{\Box}_{v,1} = \{\alpha_{n-1}\}^{\Box}_{v,1}$;

2) $\{\alpha_n\}^{\Diamond}_{v^+,0} = \{\alpha_{n-1}\}^{\Diamond}_{v^+,0}$, for every α_{n-1}–valuation v^+;

3) $\varepsilon(\{\alpha_n\}^{\Box}_{v,1}) = \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})$;

4) $\varepsilon(\{\alpha_n\}^{\Diamond}_{v^+,0}) = \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v^+,0})$, for every α_{n-1}–valuation v^+.

It follows that, for every α_{n-1}–valuation v^+,

5) if $v<n-1>v^+$ then $v<n>v^+$.

From the induction hypothesis, v is $\Box_0$– and $\Diamond_1$–α_{n-1}–normal, so we have:

6) for every p, every q, $q < p < n$ sucht that $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$], there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$ [$v_p(A_q) = 1$] and $v<n-1>v_p$.

Now, for each p, let $v_p{}^*$ be the canonical extension of v_p to α_n. Obviously $v_p{}^*(A_q) = v_p(A_q)$, and, from (†), $v_p{}^*$ is an α_n–valuation. From this, 5) and 6), then:

7) for every p, every q, $q < p < n$ sucht that $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$], there is an α_n–valuation $v_p{}^*$ such that $v_p{}^*(A_q) = 0$ [$v_p(A_q) = 1$] and $v<n>v_p{}^*$.

On the other hand, since v is an α_n–valuation, we have:

8) there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v<n-1>v_n$.

Now let $v_n{}^*$ be the canonical extension of v_n to α_n. Obviously $v_n{}^*(A_m) = v_n(A_m)$, and, from (†), $v_n{}^*$ is an α_n–valuation.

Thus we have from this fact, together with 5) and 8), and from the fact that $A_n \neq \Diamond A_m$:

9) for $p = n$, $q = m$, $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$], there is an α_n–valuation $v_p{}^*$ such that $v_p{}^*(A_q) = 0$ [$v_p(A_q) = 1$] and $v<n>v_p{}^*$.

From 7) and 9), then, v is an $\Box_0$– and $\Diamond_1$–α_n–normal.

Now, since $v(A_n) = 0$, v is trivially $\Box_1$– and $\Diamond_0$–α_n–normal (for systems other than **K**, **KB** and **K45**).

(II) Let $v(A_n) = 1$. We then have:

1) $\{\alpha_n\}^{\Box}_{v,1} = \{\alpha_{n-1}\}^{\Box}_{v,1} \cup \{A_n\}$;

2) $\{\alpha_n\}^{\Diamond}{}_{v^+,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{v^+,0}$, for every α_{n-1}–valuation v^+;

3) $\varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}) = \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}$;

4) $\varepsilon(\{\alpha_n\}^{\Diamond}{}_{v^+,0}) = \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v^+,0})$, for every α_{n-1}–valuation v^+.

Since $v(A_n) = 1$, we have from definition 1 that:

5) for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$] there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$ [$v_p(A_q) = 1$], $v_p(A_m) = 1$ and $v{<}n\text{-}1{>}v_p$.

For each p, let $v_p{}^*$ be the canonical extension of v_p to α_n. Obviously $v_p{}^*(A_q) = v_p(A_q)$, and, from (†), $v_p{}^*$ is an α_n–valuation. It follows that:

6) for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$] there is an α_n–valuation $v_p{}^*$ such that $v_p{}^*(A_q) = 0$ [$v_p{}^*(A_q) = 1$], $v_p{}^*(A_m) = 1$ and $v{<}n\text{-}1{>}v_p{}^*$.

We only need to prove now that $v{<}n{>}v_p{}^*$; the $\Box_0$– and $\Diamond_1$–α_n–normality follows. In order to do so we need to consider some logics separately.

α) **K, KD, KT**:

Since $v_p{}^*(A_m) = 1$, $v_p{}^* \vDash_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}$; thus, from 3), $v_p{}^* \vDash_1 \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})$. From 4), $v_p{}^* \vDash_0 \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0})$. Hence $v{<}n{>}v_p{}^*$, and v is $\Box_0$– and $\Diamond_1$–α_n–normal.

β) **KB, KDB, KTB**:

Since $v_p{}^*(A_m) = 1$, $v_p{}^* \vDash_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}$; thus, from 3), $v_p{}^* \vDash_1 \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})$. From 4), $v_p{}^* \vDash_0 \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})$.

Since $v{<}n\text{-}1{>}v_p{}^*$, we have by definition that $v \vDash_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v_p{}^*,1})$, $v \vDash_0 \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v_p{}^*,0})$. From 4), $v \vDash_0 \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v_p{}^*,0})$.

Now, if $v_p{}^*(\mathrm{A}_\mathrm{n}) = 0$, $\varepsilon(\{\alpha_n\}^{\Box}{}_{v_p{}^*,1}) = \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v_p{}^*,1})$, so $v \vDash_1 \varepsilon(\{\alpha_n\}^{\Box}{}_{v_p{}^*,1})$. If $v_p{}^*(A_n) = 1$, it follows from the definition of canonical extension that for every α_{n-1}–valuation v^+, if $v_p{<}n\text{-}1{>}v^+$ then $v^+(A_m) = 1$. But now, since $v{<}n\text{-}1{>}v_p{}^*$, it follows that $v_p{}^*{<}n\text{-}1{>}v$, and, since v_p and $v_p{}^*$ agree for $i < n$, $v_p{<}n\text{-}1{>}v$. Thus $v(A_m) = 1$, $v \vDash_1 \varepsilon(\{\alpha_n\}^{\Box}{}_{v_p{}^*,1})$. In any case, $v{<}n{>}v_p{}^*$; hence v is $\Box_0$– and $\Diamond_1$–α_n–normal.

γ) **KT5**:

From 2), $\{\alpha_n\}^{\Diamond}{}_{v^+,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{v^+,0}$, for every α_{n-1}–valuation v^+; so, since $v{<}n\text{-}1{>}v_p{}^*$, $\{\alpha_n\}^{\Diamond}{}_{v,0} = \{\alpha_n\}^{\Diamond}{}_{v_p{}^*,0}$.

A) If, now, $v_p{}^*(A_n) = 1$, $\{\alpha_n\}^{\Box}{}_{v_p{}^*,1} = \{\alpha_{n-1}\}^{\Box}{}_{v_p{}^*,1} \cup \{A_n\}$; $\{\alpha_n\}^{\Box}{}_{v,1} = \{\alpha_n\}^{\Box}{}_{v_p{}^*,1}$ and thus $v{<}n{>}v_p{}^*$. Hence v is $\Box_0$– and $\Diamond_1$–α_n–normal.

B) Suppose now $v_p{}^*(A_n) = 0$. We define, for every p, a new function $v_p{}^{\#}$ from FOR into $\{0,1\}$ in the following way: for every formula B,

1) if A_n is not a subformula of B, then $v_p{}^{\#}(B) = v_p{}^*(B)$;

2) if A_n is a subformula of B, then

a) for $B = A_n$, $v_p{}^{\#}(B) = 1$;
b) for $B = \neg C$, $v_p{}^{\#}(B) = 1$ iff $v_p{}^{\#}(C) = 0$;
c) for $B = C \rightarrow D$, $v_p{}^{\#}(B) = 1$ iff $v_p{}^{\#}(C) = 0$ or $v_p{}^{\#}(D) = 1$;

d) for $B = \Box C$ or $B = \Diamond C$, $v_p^{\#}(\mathrm{B}) = v_p{}^*(B)$.

It is now easy to see (with the same reasoning as in Lemma L10) that $v_p^{\#}$ is a semi-valuation. Besides, for $1 \leq i < n$, $v_p^{\#}(A_i) = v_p{}^*(A_i)$. Since $v_p{}^*$ is an α_{n-1}–valuation, $v_p^{\#}$ is an α_{n-1}–valuation. We prove that $v_p^{\#}$ is an α_n–valuation for **KT5**. First, we have that $v_p{}^*(A_m) = 1$, so $v_p^{\#}(A_m) = 1$. Let us now suppose that there is $r, s, s < r \leq n$, such that $A_r = \Box A_s$ and $v_p^{\#}(A_r) = 0$. Now, $\{\alpha_{n-1}\}^{\Box}{}_{v_p*,1} = \{\alpha_{n-1}\}^{\Box}{}_{vp\#,1}$; and $\{\alpha_{n-1}\}^{\Diamond}{}_{v_p*,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{vp\#,0}$. Since $v{<}n\text{-}1{>}v_p{}^*$, we have that $\{\alpha_{n-1}\}^{\Box}{}_{v,1} = \{\alpha_{n-1}\}^{\Box}{}_{vp\#,1}$; and $\{\alpha_{n-1}\}^{\Diamond}{}_{v,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{vp\#,0}$. Thus for every r, every $s, s < r \leq n$, if $A_r = \Box A_s$ and $v_p^{\#}(A_r) = 0$ then $v(A_r) = 0$. From D9 it follows that there is an α_{n-1}–valuation v_r such that $v_r(A_s) = 0$, $v_r(A_m) = 1$ and $v{<}n\text{-}1{>}v_r$. Thus $v_p^{\#}{<}n\text{-}1{>}v_r$, and $v_p^{\#}$ is an α_n–valuation for **KT5**. Now, since $v_p^{\#}(A_n) = 1$, $\{\alpha_n\}^{\Box}{}_{v,1} = \{\alpha_n\}^{\Box}{}_{vp\#,1}$; and $\{\alpha_n\}^{\Diamond}{}_{v,0} = \{\alpha_n\}^{\Diamond}{}_{vp\#,0}$. Thus $v{<}n{>}v_p^{\#}$ and it follows that for every p, every $q, q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$ there is an α_n–valuation $v_p^{\#}$ such that $v_p^{\#}(A_q) = 0$, $v_p^{\#}(A_m) = 1$ and $v{<}n{>}v_p^{\#}$. If now there is $p, q, q < p \leq n$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, the proof is similar. That is, v is $\Box_0$– and $\Diamond_1$–α_n–normal.

δ) **K45, KD45**:

We prove as in β) that $v_p{}^* \vDash_1 \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})$, $v_p{}^* \vDash_0 \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})$. From 2), $\{\alpha_n\}^{\Diamond}{}_{v+,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{v+,0}$, for every α_{n-1}–valuation v^+; so, since $v{<}n\text{-}1{>}v_p{}^*$, $\{\alpha_n\}^{\Diamond}{}_{v,0} = \{\alpha_n\}^{\Diamond}{}_{v_p*,0}$.

A) If, now, $v_p{}^*(A_n) = 1$, $\{\alpha_n\}^{\Box}{}_{v_p*,1} = \{\alpha_{n-1}\}^{\Box}{}_{v_p*,1} \cup \{A_n\}$; $\{\alpha_n\}^{\Box}{}_{v,1} = \{\alpha_n\}^{\Box}{}_{v_p*,1}$. Thus $v{<}n{>}v_p{}^*$, and v is $\Box_0$– and $\Diamond_1$–α_n–normal.

B) Suppose now $v_p{}^*(\mathrm{A_n}) = 0$. We define as in γ), for every p, a new function $v_p^{\#}$ from FOR into $\{0,1\}$. It is now easy to see (with the same reasoning as in L10) that $v_p^{\#}$ is a semi-valuation. Besides, for $1 \leq i < n$, $v_p^{\#}(A_i) = v_p{}^*(A_i)$. Since $v_p{}^*$ is an α_{n-1}–valuation, $v_p^{\#}$ is an α_{n-1}–valuation. We prove that $v_p^{\#}$ is an α_n–valuation for **K45** and **KD45**. First, we have that $v_p{}^*(A_m) = 1$, so $v_p^{\#}(A_m) = 1$. Let us now suppose that there is $r, s, s < r \leq n$, such that $A_r = \Box A_s$ and $v_p^{\#}(A_r) = 0$. Now, $\{\alpha_{n-1}\}^{\Box}{}_{v_p*,1} = \{\alpha_{n-1}\}^{\Box}{}_{vp\#,1}$; and $\{\alpha_{n-1}\}^{\Diamond}{}_{v_p*,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{vp\#,0}$. Since $v{<}n\text{-}1{>}v_p{}^*$, we have that $\{\alpha_{n-1}\}^{\Box}{}_{v,1} = \{\alpha_{n-1}\}^{\Box}{}_{vp\#,1}$; and $\{\alpha_{n-1}\}^{\Diamond}{}_{v,0} = \{\alpha_{n-1}\}^{\Diamond}{}_{vp\#,0}$. It also follows, thus, that $\varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) = \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{vp\#,1})$; and $\varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}) = \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{vp\#,0})$. Thus for every r, every $s, s < r \leq n$, if $A_r = \Box A_s$ and $v_p^{\#}(A_r) = 0$ then $v(A_r) = 0$. From definition 1 it follows that there is an α_{n-1}–valuation v_r such that $v_r(A_s) = 0$, $v_r(A_m) = 1$ and $v{<}n\text{-}1{>}v_r$. Thus $v_p^{\#}{<}n\text{-}1{>}v_r$, and $v_p^{\#}$ is an α_n–valuation for **K45** and **KD45**. Now, since $v_p^{\#}(A_n) = 1$, $\{\alpha_n\}^{\Box}{}_{v,1} = \{\alpha_n\}^{\Box}{}_{vp\#,1}$; and $\{\alpha_n\}^{\Diamond}{}_{v,0} = \{\alpha_n\}^{\Diamond}{}_{vp\#,0}$. Thus $v{<}n{>}v_p^{\#}$ and it follows that for every p, every $q, q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$ there is an α_n–valuation $v_p^{\#}$ such that $v_p^{\#}(A_q) = 0$, $v_p^{\#}(A_m) = 1$ and $v{<}n{>}v_p^{\#}$. If now there is $p, q, q < p \leq n$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, the proof is similar. That is, v is $\Box_0$– and $\Diamond_1$–α_n–normal.

We prove now that v is $\Box_1$– and $\Diamond_0$–α_n–normal (for the systems different from **K**, **KB** and **K45**, of course). That v is $\Diamond_0$–α_n–normal follows trivially from the fact that it is $\Diamond_0$–α_{n-1}–normal, because $A_n \neq \Diamond A_m$. By induction hypothesis, v is $\Box_1$–α_{n-1}–normal, and, from D9, we have that $v(A_m) = 1$ (for **KT**, **KTB** and **KT5**). That is, v is $\Box_1$–α_n–normal. In the case of **KD**, **KDB** and **KD45**, from D9, for $p = n$, $q = m$, there is an α_{n-1}–valuation v_p such that $v_p(A_m) = 1$ and $v{<}n\text{-}1{>}v_p$. We take the canonical extension $v_p{}^*$ from v_p to α_n. It is of course an α_n–valuation, and, since $v_p{}^*(A_m) = 1$, it follows from 3) and 4) that $v{<}n{>}v_p{}^*$. So v is $\Box_1$–α_n–normal.

(3) Let us suppose that, for some $m < n$, $A_n = \Diamond A_m$. Proof as in (2). ■

As a direct result of combining this Lemma with Proposition P20, we have the following

Corollary. *Let α_n be a normal sequence, v an α_{n-1}–valuation and v_c the canonical extension of v to α_n. Then v_c is an α_n–valuation and, for $1 \leq i \leq n-1$, $v_c(A_i) = v(A_i)$.*

That is, now we are sure, if we have some α_n–valuation, and if we build a normal sequence $A_1,\ldots,A_n,A_{n+1},\ldots,A_s$, that it is always possible to extend this α_n–valuation to the new sequence. The next theorem puts all these fact together:

Theorem T18. *v is an α_n–valuation iff: 1) α_n is a normal sequence; 2) v is a semi–valuation; 3) v is α_n–normal.*

Proof. Immediate. ■

3.4 Correctness

Having thus proved these properties of (α_n–)valuations, we are now ready to consider correctness. The notions of satisfiability, validity and semantical consequence are defined, as one could expect, in the standard way: a formula A is *satisfiable* if there is some valuation v such that $v(A) = 1$. A is *valid* ($\models A$) if for ali valuations v, v satisfies A. Last but not least, if Γ is a set of wffs, we say that A is a *semantical consequence of* Γ, or that Γ *semantically implies* A ($\Gamma \models A$), if, for every valuation v such that $v \models \Gamma$, $v(A) = 1$. ("$v \models \Gamma$", of course, means that $v(B) = 1$, for ali $B \in \Gamma$. And, needless to say, all this is relevant to some logic L.)

In the following, let L be one of **K**, **KB**, **K45**, **KD**, **KDB**, **KD45**, **KT**, **KTB**, **KT5**.

Lemma L12. *Let v be an α_n–valuation; then, for $1 \leq i \leq n$, if A_i is an axiom of L then $v(A_i) = 1$.*

Proof. If A_i is an axiom of one of the said logics, then it is either an axiom from **PL**, and it follows from the fact that v is a semi–valuation, that $v(A_i) = 1$, or it is one of the modal axiom schemes. We consider each case.

(a) $A_i = \Diamond A \leftrightarrow \neg\Box\neg A$. Suppose $v(\Diamond A \leftrightarrow \neg\Box\neg A) = 0$. Then we have, say, $v(\Diamond A) = 1$ and $v(\neg\Box\neg A) = 0$, so $v(\Box\neg A) = 1$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, there is an α_n–valuation v_p such that $v_p(A_q) = 1$ and $v{<}n{>}v_p$. Thus $v_p(\mathrm{A}) = 1$. Now we consider each logic:

a) **K**, **KB**, **K45**, **KD**, **KDB**, **KD45**, **KT**, **KTB**:
$v{<}n{>}v_p$ means (among other things) that $v_p \models_1 \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})$. Hence $v_p(\neg A) = 1$, $v_p(A) = 0$, what cannot be, since we already had $v_p(A) = 1$.

b) **KT5**:

$v{<}n{>}v_p$ means (among other things) that $v_p \models_1 \{\alpha_n\}^{\Box}_{v,1}$. Hence $v_p(\Box\neg A) = 1$. Now, v_p is $\Box_1$–α_n–normal, so $v_p(\neg A) = 1$, $v_p(A) = 0$, what cannot be.

If now $v(\Diamond A) = 0$ and $v(\neg\Box\neg A) = 1$, the proof goes in a similar way. Hence $v(\Diamond A \leftrightarrow \neg\Box\neg A) = 1$.

(b) $A_i = \Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$. Suppose $v(A_i) = 0$. Then we have $v(\Box(A \rightarrow B)) = v(\Box A) = 1$ and $v(\Box B) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p such that $v_p(A_q) = 0$ and $v{<}n{>}v_p$. Thus $v_p(B) = 0$. Now we consider the logics in two cases:

a) **K, KB, K45, KD, KDB, KD45, KT, KTB**:
$v{<}n{>}v_p$ means (among other things) that $v_p \models_1 \varepsilon(\{\alpha_n\}^{\Box}_{v,1})$. Hence $v_p(A) = v_p(A \rightarrow B) = 1$, $v_p(B) = 0$, what cannot be, since v_p is also a semi-valuation.

b) **KT5**:
$v{<}n{>}v_p$ means (among other things) that $v_p \models_1 \{\alpha_k\}^{\Box}_{v,1}$. Hence $v_p(\Box(A \rightarrow B)) = v_p(\Box A) = 1$ Now, v_p is $\Box_1$–α_n–normal, so $v_p(A) = v_p(A \rightarrow B) = 1$, $v_p(B) = 0$, what cannot be.
Hence $v(A_i) = 1$.

We must now consider the special axioms of each system.

(c) $A_i = \Box A \rightarrow \Diamond A$. **(KD, KDB, KD45)** Suppose $v(A_i) = 0$. Then we have $v(\Box A) = 1$ and $v(\Diamond A) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 1$, there is an α_n–valuation v_p such that $v_p(A_q) = 1$ and $v{<}n{>}v_p$. Thus $v_p(A) = 1$. Now $v{<}n{>}v_p$ means that $v_p \models_0 \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})$. Hence $v_p(A) = 0$, a contradiction. Thus $v(A_i) = 1$.

(d) $A_i = \Box A \rightarrow A$. **(KT, KTB, KT5)** Suppose $v(A_i) = 0$. Then we have $v(\Box A) = 1$ and $v(A) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 1$, $v(A_q) = 1$. Thus $v(A) = 1$; a contradiction. Hence $v(A_i) = 1$.

(e) $A_i = A \rightarrow \Box\Diamond A$. **(KTB)** Suppose $v(A_i) = 0$. Then we have $v(A) = 1$ and $v(\Box\Diamond A) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p such that $v_p(A_q) = 0$ and $v{<}n{>}v_p$. Thus $v_p(\Diamond A) = 0$. Now $v{<}n{>}v_p$ means (among other) that $v \models_0 \varepsilon(\{\alpha_n\}^{\Diamond}_{v_p,0})$. Hence $v(A) = 0$, what cannot be; thus $v(A_i) = 1$.

(f) $A_i = \Box A \rightarrow \Box\Box A$. **(K45, KD45)** Suppose $v(A_i) = 0$. Then we have $v(\Box A) = 1$ and $v(\Box\Box A) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p such that $v_p(A_q) = 0$ and $v{<}n{>}v_p$. Thus $v_p(\Box A) = 0$. Now $v{<}n{>}v_p$ means (among other) that $\{\alpha_n\}^{\Box}_{v,1} = \{\alpha_n\}^{\Box}_{v_p,1}$. Hence $v(\Box A) = 0$, what cannot be; thus $v(A_i) = 1$.

(g) $A_i = \Diamond A \rightarrow \Box\Diamond A$. **(K45, KD45, KT5)** Suppose $v(A_i) = 0$. Then we have $v(\Diamond A) = 1$ and $v(\Box\Diamond A) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p such that $v_p(A_q) = 0$ and $v{<}n{>}v_p$. Thus $v_p(\Diamond A) = 0$. Now $v{<}n{>}v_p$ means (among other) that $\{\alpha_n\}^{\Diamond}_{v,0} = \{\alpha_n\}^{\Diamond}_{v_p,0}$. Hence $v(\Diamond A) = 0$, what cannot be; thus $v(A_i) = 1$. ■

Theorem T19. *If A is an axiom of L and v is a valuation, then $v(A) = 1$.*

Proof. Let A be an axiom of one of the said logics, and v a valuation. Let α_n be a normal sequence such that, for some $i \leq n$, $A = A_i$. By definition, v is an α_n–valuation and, from L12, $v(A) = 1$.■

Lemma L13. *For all n, all i, $1 \leq i \leq n$, if v is an α_n–valuation and $\vdash_L A_i$, then $v(A_i) = 1$.*

Proof. By induction on the number r of lines of a proof of A_i in L.

A) $r = 1$. Then A_i is an axiom, and the property follows from L12.

B) $r > 1$. If A_i is an axiom, the property follows from L12; else:

(a) A_i was obtained by *MP* from B and $B \rightarrow A_i$. We have that $\vdash B$ and $\vdash B \rightarrow A_i$. Let us form the following set $\tau = \{C : C$ is a subformula of $B \rightarrow A_i$ and $C \notin \{A_1,...,A_n\}\}$. If $\tau \neq \emptyset$, let us put the elements of τ in a sequence $C_1,...,C_k$ respecting the length of the formulas. If $\tau = \emptyset$, let $\sigma = A_1,...,A_n$ and $v_c = v$. Else let $\sigma = A_1,...,A_n,C_1,...,C_k$, and let us define a sequence $v_0,v_1,...,v_k$ where $v_0 = v$ and, for $1 \leq j \leq k$, let v_j be the canonical extension of v_{j-1}. Let us take $v_c = v_k$. Obviously σ is a normal sequence, and v_c a σ–valuation. Since $\vdash B$ and $\vdash B \rightarrow A_i$, we have by the induction hypothesis that $v_c(B) = v_c(B \rightarrow A_i) = 1$. So $v_c(A_i) = 1$. Since $v(A_i) = v_c(A_i)$, $v(A_i) = 1$.

(b) $A_i = \Box B$ and was obtained by *RN* from B. Well, in every normal sequence σ in which A_i occurs, B occurs too; so, by the induction hypothesis, for every σ–valuation v, $v(B) = 1$. If now $v(A_i)$ were to be 0, there should be an α_n–valuation v_n such that $v_n(B) = 0$, what cannot be. So $v(A_i) = 1$. ■

Corollary. *If $\vdash A$ then $\vDash A$.*

Proof. Suppose $\vdash A$, and let v be a valuation. Let α_n be a normal sequence in which, for some $i \leq n$, $A = A_i$. By definition, v is an α_n–valuation, so $v(A) = 1$ from L12, and thus $\vDash A$. ■

Theorem T20. (*Correctness Theorem*) *If $\Gamma \vdash A$ then $\Gamma \vDash A$.*

Proof. Suppose $\Gamma \vdash A$, and let $D_1,...,D_r$ be a deduetion of A from Γ. We prove the theorem by induction on r.

A) $r = 1$. Then, either $A \in \Gamma$, and we have nothing to prove, or A is an axiom, so A is valid (by corollary to L13 and $\Gamma \vDash A$.

B) $r > 1$. If $A \notin \Gamma$ and A is not an axiom, then:

(a) for some $j < r$, $i < r$, $D_i = D_j \rightarrow A$. So $\Gamma \vdash D_i$, $\Gamma \vdash D_i \rightarrow A$ and, by the induction hypothesis, $\Gamma \vDash D_i$, $\Gamma \vDash D_i \rightarrow A$. Thus, for every valuation v, if $v \vDash \Gamma$, $v(D_j) = v(D_j \rightarrow A) = 1$, and hence $v(A) = 1$. So $\Gamma \vDash A$.

(b) $A = \Box B$ and, for some $j < r$, $D_r = B$. In this case, $\vdash D_r$ and $\vdash A$. By the Corollary to L13, for every valuation v, $v(A) = 1$. So, if $v \vDash \Gamma$, $v(A) = 1$. Thus $\Gamma \vDash A$. ■

3.4 Completeness

Completeness is now easy to prove making use of saturated sets—which are just MCSs. They are defined in the very same way as in chapter 1, thus properties like in P5 or P6 also hold:

Proposition P21. *If Δ is saturated, then:*

(a) $A \in \Delta$ *iff* $\Delta \vdash A$;
(b) $\neg A \in \Delta$ *iff* $A \notin \Delta$;
(c) $A \rightarrow B \in \Delta$ *iff* $A \notin \Delta$ *or* $B \in \Delta$.

Proposition P22. *If $\Gamma \nvdash A$, then there is an A-saturated set Δ such that $\Gamma \subseteq \Delta$.*

We now consider some properties that we'll need in the completeness proof.

Lemma L14. *Let Δ, Θ be any saturated sets of wffs, Γ any set of wffs. Following properties hold:*

a) L (that is, every normal logic):

i) *If $\Gamma \nvdash \Box A$, then there is an A-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.*
ii) *If $\Gamma \nvdash \neg\Diamond A$, then there is an $\neg A$-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.*

b) **KD, KDB, KD45**:

i) *If $\Gamma \nvdash \neg\Box A$, then there is an $\neg A$-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.*
ii) *If $\Gamma \nvdash \Diamond A$, then there is an A–saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.*

c) **KT, KTB, KT5**:

i) $\varepsilon(\Delta^{\Box}) \subseteq \Delta$;
ii) $\varepsilon(\Delta^{\neg\Diamond}) \subseteq \Delta$;

d) **KB, KDB, KTB, KT5**:

i) $\varepsilon(\Delta^{\Box}) \subseteq \Theta$ *iff* $\varepsilon(\Theta^{\Box}) \subseteq \Delta$;
ii) $\varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$ *iff* $\varepsilon(\Theta^{\neg\Diamond}) \subseteq \Delta$;

e) **K45, KD45, KT5**:

i) *if $\varepsilon(\Delta^{\Box}) \cup \varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$ then $\Delta^{\Box} = \Theta^{\Box}$ and $\Delta^{\neg\Diamond} = \Theta^{\neg\Diamond}$;*
ii) $\Delta^{\Box} \subseteq \varepsilon(\Delta^{\Box})$;
iii) $\Delta^{\neg\Diamond} \subseteq \varepsilon(\Delta^{\neg\Diamond})$.

Proof. (a.i) Suppose $\Gamma \nvdash \Box A$. Then $\Gamma^{\Box} \cup \Gamma^{\neg\Diamond} \nvdash \Box A$, since both are subsets of Γ. By P19, $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \nvdash A$. From P21, there is an A-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.

(a.ii) If $\Gamma \nvdash \neg\Diamond A$, then $\Gamma \nvdash \Box\neg A$. By (a.i) there is a $\neg A$-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.

(b.i) Suppose $\Gamma \nvdash \neg\Box A$. Since *Df*$\Diamond$ and D are axioms of these logics, we have that $\Box\neg A \rightarrow \neg\Box A$ as theorem. So $\Gamma \nvdash \Box\neg A$. By (a.i) there is a $\neg A$-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.

(b.ii) Suppose $\Gamma \nvdash \Diamond A$. Since D is an axiom of these logics, $\Gamma \nvdash \Box A$. By (a.i) there is a A-saturated set Δ such that $\varepsilon(\Gamma^{\Box}) \cup \varepsilon(\Gamma^{\neg\Diamond}) \subseteq \Delta$.

(c.i) Let $A \in \varepsilon(\Delta^{\Box})$. So $\Box A \in \Delta$. Since T is an axiom, $A \in \Delta$.

(c.ii) Let $\neg A \in \varepsilon(\Delta^{\neg\Diamond})$. So $\neg\Diamond A \in \Delta$. Since T is an axiom, $A \rightarrow \Diamond A$ is a theorem, so $A \notin \Delta$; $\neg A \in \Delta$.

(d.i) Suppose $\varepsilon(\Delta^{\Box}) \subseteq \Theta$, and let $A \in \varepsilon(\Theta^{\Box})$. So $\Box A \in \Theta$; $\neg\Box A \notin \Theta$ and $\Box\neg\Box A \notin \Delta$. But then $\neg\Box\neg\Box A \in \Delta$; that is, $\Diamond\Box A \in \Delta$, and, since $\vdash \Diamond\Box A \rightarrow A$, $A \in \Delta$. The other direction and (d.ii) are similar.

(e.i) Suppose $\varepsilon(\Delta^{\Box}) \cup \varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$, and let $\Box A \in \Delta^{\Box}$. Thus $\Box A \in \Delta$ and, since *4* is an axiom, $\Box\Box A \in \Delta$. Thus $\Box A \in \Theta$; $\Box A \in \Theta^{\Box}$. Let now $\Box A \in \Theta^{\Box}$. Then $\Box A \in \Theta$. If $\Box A \notin \Delta^{\Box}$, $\Box A \notin \Delta$; $\neg\Box A \in \Delta$. Since *5* is an axiom we have as theorem $\neg\Box A \rightarrow \Box\neg\Box A$. So $\Box\neg\Box A \in \Delta$. But then $\neg\Box A \in \Theta$, because $\varepsilon(\Delta^{\Box}) \subseteq \Theta$, and this is a contradiction. So $\Box A \in \Delta^{\Box}$. Let now $\neg\Diamond A \in \Delta^{\neg\Diamond}$. Thus $\neg\Diamond A \in \Delta$ and, since *4* is an axiom, $\neg\Diamond\Diamond A \in \Delta$. Since $\varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$, we have that $\neg\Diamond A \in \Theta$; $\neg\Diamond A \in \Theta^{\neg\Diamond}$. Let now $\neg\Diamond A \in \Theta^{\neg\Diamond}$. Then $\neg\Diamond A \in \Theta$. If $\neg\Diamond A \notin \Delta^{\neg\Diamond}$; $\neg\Diamond A \notin \Delta$; $\Diamond A \in \Delta$. Since *5* is an axiom, we have as theorem $\Diamond\neg\Diamond A \rightarrow \neg\Diamond A$. So $\neg\Diamond\neg\Diamond A \in \Delta$; $\Box\Diamond A \in \Delta$; and thus $\Diamond A \in \Theta$, since $\varepsilon(\Delta^{\Box}) \subseteq \Theta$. But this cannot be, so If $\neg\Diamond A \in \Delta^{\neg\Diamond}$.

(e.ii) If $\Box A \in \Delta$, then, since *4* is an axiom, $\Box\Box A \in \Delta$, $\Box A \in \varepsilon(\Delta^{\Box})$. (e.iii) is proved in a similar way. ■

Theorem T21. *For every A–saturated set Δ and every normal sequence α_n, the characteristic function f of Δ is an α_n–valuation.*

Proof. First of ali, it is easy to prove by P21 that

(†) The characteristic function f of Δ is a semi–valuation.

We now prove the theorem by induction on n. If $n = 1$, the property follows from (†) above. Let us suppose $n > 1$.

(1) If, for every $m < n$, $A_n \neq \Box A_m$, $A_n \neq \Diamond A_m$, f is trivially an α_n–valuation.

(2) For some $m < n$, $A_n = \Box A_m$.

(I) $f(A_n) = 0$. Then $A_n \notin \Delta$, $\Delta \nvdash \Box A_m$. From L13, there is an A_m-saturated set Θ such that $\varepsilon(\Delta^{\Box}) \cup \varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$. Let f_Θ be the characteristic function of Θ. By the induction hypothesis, f and f_Θ are α_{n-1}–valuations. We also have, since Θ is A_m-saturated, that $f_\Theta(A_m) = 0$. We consider now each logic:

α) **K, KD and KT:**
Now, $\{\alpha_{n-1}\}^{\Box}_{f,1} \subseteq \Delta$, thus $\varepsilon(\{\alpha_{n-1}\}^{\Box}_{f,1}) \subseteq \varepsilon(\Delta^{\Box}) \subseteq \Theta$; thus $f_\Theta \models_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}_{f,1})$. Let now $\Diamond A \in \{\alpha_{n-1}\}^{\Diamond}_{f,0}$. Then $f_\Theta(\neg\Diamond A) = 1$, so $f_\Theta(\neg A) = 1$, $f_\Theta(A) = 0$. Thus $f_\Theta \models_0 \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{f,0})$. We can thus say that $f{<}n\text{-}1{>}f_\Theta$; hence f is an α_n–valuation.

β) **KB, KDB, KTB:**
We prove as in α) that $f_\Theta \models_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}_{f,1})$ and $f_\Theta \models_0 \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{f,0})$. Now, from L13, $\varepsilon(\Delta^{\Box}) \subseteq \Theta$ iff $\varepsilon(\Theta^{\Box}) \subseteq \Delta$; and $\varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$ iff $\varepsilon(\Theta^{\neg\Diamond}) \subseteq \Delta$. It is easy to conclude that $f \models_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}_{f_\Theta,1})$ and $f \models_0 \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{f_\Theta,0})$. We thus can say that $f{<}n\text{-}1{>}f_\Theta$; hence f is an $\Delta_1,\ldots,A_n$–valuation.

γ) **KT5:**
We prove as in α) that $f_\Theta \models_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}_{f,1})$ and $f_\Theta \models_0 \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{f,0})$, and as in β) that $f \models_1 \varepsilon(\{\alpha_{n-1}\}^{\Box}_{f_\Theta,1})$ and $f \models_0 \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{f_\Theta,0})$. Now, from L13, $\Delta^{\Box} \subseteq \varepsilon(\Delta^{\Box})$ and $\Delta^{\neg\Diamond} \subseteq \varepsilon(\Delta^{\neg\Diamond})$. It is then easy to see that $\{\alpha_{n-1}\}^{\Box}_{f,1} = \{\alpha_{n-1}\}^{\Box}_{f_\Theta,1}$, and $\{\alpha_{n-1}\}^{\Diamond}_{f,0} = \{\alpha_{n-1}\}^{\Diamond}_{f_\Theta,0}$. We thus can say that $f{<}n\text{-}1{>}f_\Theta$; hence f is an α_n–valuation.

δ) **K45, KD45:**

We prove as in α) that $f_\Theta \vDash_1 \varepsilon(\{\alpha_{n\text{-}1}\}^{\Box}{}_{f,1})$ and $f_\Theta \vDash_0 \varepsilon(\{\alpha_{n\text{-}1}\}^{\Diamond}{}_{f,0})$, and it is easy to see, from (e.i) of L13, that $\{\alpha_{n\text{-}1}\}^{\Box}{}_{f,1} = \{\alpha_{n\text{-}1}\}^{\Box}{}_{f_\Theta,1}$, and $\{\alpha_{n\text{-}1}\}^{\Diamond}{}_{f,0} = \{\alpha_{n\text{-}1}\}^{\Diamond}{}_{f_\Theta,0}$. We thus can say that $f{<}n\text{-}1{>}f_\Theta$; hence f is an α_n–valuation.

(II) $f(A_n) = 1$. So $\Box A_m \in \Delta$, $\Delta \vdash \Box A_m$. Let us suppose there is some p, some q, $q < p \leq n$ such that $A_p = \Box A_q$ and $f(A_p) = 0$. From L13, there is an A_q–saturated set Θ such that $\varepsilon(\Delta^{\Box}) \cup \varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$. Let f_Θ be the characteristic function of Θ. By the induction hypothesis, f and f_Θ are $\alpha_{n\text{-}1}$–valuations. With an analogous argument as in case I), we show that $f{<}n\text{-}1{>}f_\Theta$. Since Θ is A_q–saturated, $f_\Theta(A_q) = 0$ and, since $A_m \in \varepsilon(\Delta^{\Box})$, $f_\Theta(A_m) = 1$.
Now, in the case of **KD**, **KDB**, **KD45** it follows from L13, since $\Delta \vdash \Box A_m$, that there is an $\neg A_m$–saturated set Θ such that $\varepsilon(\Delta^{\Box}) \cup \varepsilon(\Delta^{\neg\Diamond}) \subseteq \Theta$. We prove in a similar way that f_Θ is an $\alpha_{n\text{-}1}$–valuation, $f{<}n\text{-}1{>}f_\Theta$ and $f_\Theta(A_m) = 1$.
In the case of **KT**, **KTB**, **KT5**, it follows from L13, since $\Delta \vdash \Box A_m$, that $A_m \in \Delta$. So $f(A_m) = 1$.
If there is now some p, some q, $q < p \leq n$ such that $A_p = \Diamond A_q$ and $f(A_p) = 1$, the proof is similar. It follows that f is an α_n–valuation.

(3) For some $m < n$, $A_n = \Diamond A_m$. Proof as in (2). ∎

With this result we come now to the following

Corollary. *v is a valuation iff v is the characteristic function of some saturated set Δ.*

Proof (a) Let us suppose that v is a valuation. Let $[v]_1 = \{A : v(A) = 1\}$, and let $[v]_0 = \{B : v(B) = 0\}$. Let $C \in [v]_0$; so $C \notin [v]_1$ and we easily see that $[v]_1 \nvdash C$. Let D be a formula such that $D \notin [v]_1$. So $v(v) = 0$, $v(\neg D) = 1$ and $\neg D \in [v]_1$. But, since $\vdash \neg D \to (D \to C)$, $v(\neg D \to (D \to C)) = 1$; thus $\neg D \to (D \to C) \in [v]_1$. So $[v]_1 \vdash D \to C$, and, obviously, $[v]_1 \cup \{D\} \vdash C$. Let $\Delta = [v]_1$. Hence Δ is a C-saturated set and, by construction, v is its characteristic function.

(b) Suppose that, for some saturated set Δ, v is the characteristic function of Δ. Let α_n be any normal sequence: then (by T21) v is an α_n–valuation. Since α_n is any normal sequence, v is a valuation. ∎

Theorem T22. (*Completeness Theorem*) *If $\Gamma \vDash A$ then $\Gamma \vdash A$.*

Proof. Suppose $\Gamma \vDash$ A, and $\Gamma \nvdash$ A. Then there is an A–saturated set Δ such that $\Gamma \subseteq \Delta$. Let v be the characteristic function of Δ. By the corollary to theorem 3, v is a valuation. Since $\Gamma \subseteq \Delta$, $v \vDash \Gamma$; since Δ is A–saturated, $v(A) = 0$. So $\Gamma \nvDash A$, against the hypothesis. Hence $\Gamma \vdash A$. ∎

4

Valuation semantics for classical modal logics

I have yet to see any problem, however complicated,
which, when looked at in the right way, did not become
still more complicated.

POUL ANDERSON.

In this chapter we are then going to consider valuation semantics for classical modal logics.

A system of modal logic is called *classical* if it contains $Df\Diamond$ (i.e., $\Diamond A \leftrightarrow \neg\Box\neg A$), and if it is closed under *RE*:

$$RE: \ \vdash A \leftrightarrow B \ \ / \ \vdash \Box A \leftrightarrow \Box B.$$[42]

The smallest classical modal logic is called **E**. To name other classical systems we write, as usual, $\mathbf{ES_1 ... S_n}$ to mean the extension of **E** through axiom schemas $S_1,...,S_n$. The axiom schemas which we will be using here are the following:

$$M. \ \Box(A \wedge B) \rightarrow \Box A \wedge \Box B$$
$$C. \ \Box A \wedge \Box B \rightarrow \Box(A \wedge B)$$

Also at our disposal is the inference rule *RN* ($\vdash A \,/\, \vdash \Box A$), already known from the normal modal logics. Thus **ECN** will mean the logic obtained by adding *C* as axiom schema and *RN* as inference rule. Using all possible combinations of these axiom schemas and rule of inference, we arrive at the following picture (cf. [Ch80], p. 237) of 8 non-equivalent logics (an arrow means that the logic on the arrow's left is a subsystem of the one on the right):

$$\begin{tikzcd}
 & \mathbf{EM} \arrow[r] \arrow[dr] & \mathbf{EMC} \arrow[dr] & \\
\mathbf{E} \arrow[ur] \arrow[r] \arrow[dr] & \mathbf{EC} \arrow[ur] \arrow[dr] & \mathbf{EMN} \arrow[r] & \mathbf{EMCN} \\
 & \mathbf{EN} \arrow[ur] \arrow[r] & \mathbf{ECN} \arrow[ur] &
\end{tikzcd}$$

fig. 18

[42] More about classical modal logics can be found in Chellas [1980], chapter 8, which I am going closely to follow.

The logic **EM** is also called *M* in [Ch80], because it is the smallest *monotonic* modal logic, and **EMC** has the denomination ***R*** too, because it is the smallest *regular* modal logic.[43] The system **EMCN**, by the way, is the same **K** which we already knew—that is, the smallest normal modal logic.

Now in the case of classical logics, things work in a similar way to the normal modal logics: so we have normal sequences and semi-valuations as before. Differences are, of course, to be expected in the definitions of $A_1,...,A_n$–valuations.

5.1 Defining $A_1,...,A_n$-valuations for classical logics

As usual, we need some definitions and abbreviations. Let α_n be a normal sequence, and let us suppose that α_n–valuations, and valuations *simpliciter*, were already defined. We introduce the following abbreviations (for $1 \leq k \leq n$, and where $\Gamma, \Delta \subseteq \{\alpha_k\}$):

$A \approx_k B$	iff	for every α_k–valuation v, $v(A) = v(B)$;
$A \not\approx_k B$	iff	for every α_k–valuation v, $v(A) \neq v(B)$;
$A \approx>_k B$	iff	for every α_k–valuation v, if $v(A) = 1$ then $v(B) = 1$;
$A \not\approx>_k B$	iff	for every α_k–valuation v, if $v(A) = 1$ then $v(B) = 0$;
$A <\not\approx_k B$	iff	for every α_k–valuation v, if $v(A) = 0$ then $v(B) = 1$;
$A \approx_k <\Gamma, \Delta>$	iff	for every α_k–valuation v, $v(A) = 1$ iff $v \models_1 \Gamma$ and $v \models_0 \Delta$;
$A \not\approx_k <\Gamma, \Delta>$	iff	for every α_k–valuation v, $v(A) = 0$ iff $v \models_0 \Gamma$ and $v \models_1 \Delta$;
$<\Gamma, \Delta> \approx>_k A$	iff	for every α_k–valuation v, if $v \models_1 \Gamma$ and $v \models_0 \Delta$ then $v(A) = 1$;
$<\Gamma, \Delta> \not\approx>_k A$	iff	for every α_k–valuation v, if $v \models_0 \Gamma$ and $v \models_1 \Delta$ then $v(A) = 0$;

Now the following abbreviations, as in the case of "$f<k>g$" in normal logics, will be meaning different things for the several systems. Let α_n be a normal sequence, and let us suppose again that α_n–valuations were already defined. We introduce the following abbreviations (for $1 \leq k \leq n$, and where $\Gamma, \Delta \subseteq \{\alpha_k\}$):

(a) for **E**, **EN**:

$\xi^k[A, \Gamma] =_{df} \{ B \in \Gamma : B \approx_k A \}$;
$\chi^k[A, \Gamma] =_{df} \{ B \in \Gamma : B \not\approx_k A \}$;

(b) for **EM**, **EMN**:

$\xi^k[A, \Gamma] =_{df} \{ B \in \Gamma : B \approx>_k A \}$;
$\chi^k[A, \Gamma] =_{df} \{ B \in \Gamma : B <\not\approx_k A \}$;
$\zeta^k[A, \Gamma] =_{df} \{ B \in \Gamma : A \not\approx>_k B \}$;

[43] A system of modal logic is said to be *monotonic* iff it contains Df◇ and is closed under *RM*, i.e.: $A \rightarrow B \ / \Box A \rightarrow \Box B$. A modal logic is said to be *regular* iff it contains Df◇ and is closed und ***RR***, that is: $A \wedge B \rightarrow C \ / \Box A \wedge \Box B \rightarrow \Box C$. (Cf. [Ch80], p. 234.)

$$\eta^{k}[A, \Gamma] \;=_{df}\; \{\, B \in \Gamma : A \not\approx>_{k} B \,\};$$

(c) for **EC**, **ECN**:

$$\xi^{k}[A, \Gamma, \Delta] \;=_{df}\; \{\, <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } A \approx_{k} <\Theta, \Phi> \};$$
$$\chi^{k}[A, \Gamma, \Delta] \;=_{df}\; \{\, <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } A \not\approx_{k} <\Theta, \Phi> \};$$

(d) for **EMC**, **EMCN**:

$$\xi^{k}[A, \Gamma, \Delta] \;=_{df}\; \{\, <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } <\Theta, \Phi> \approx>_{k} A \};$$
$$\chi^{k}[A, \Gamma, \Delta] \;=_{df}\; \{\, <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } <\Theta, \Phi> \not\approx>_{k} A \}.$$

Before we go into the details of what ali these definitions mean, let us take a look at the definition of an α_n–valuation. Perhaps things will be clearer by then. I'll give first the definitions for **E**, and then we'll see which changes are needed for the other logics.

Definition D11. v is an α_n–valuation (for **E**) if α_n is a normal sequence and

1) $n = 1$ and v is a semi–valuation;

2) $n > 1$, v is an α_{n-1}–valuation and, if for some $m < n$,

A) $A_n = \Box A_m$,

I) if $v(A_n) = 0$ then $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

II) if $v(A_n) = 1$ then for every p, every q, $q < p \leq n$, such that:

a. $A_p = \Box A_q$ and $v(A_p) = 0$, $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

b. $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

B) $A_n = \Diamond A_m$,

I) if $v(A_n) = 1$ then $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

II) if $v(A_n) = 0$ then for every p, every q, $q < p \leq n$, such that:

a. $A_p = \Box A_q$ and $v(A_p) = 0$, $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}) \cup \{A_m\}] = \emptyset$;

b. $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}) \cup \{A_m\}] = \emptyset$.

This definition, too, looks scaring, but by now the reader has probably got a feeling of how things work with valuation semantics. Everything is like in the normal logics case, but the way we treat the modal operators. Let us consider the case where $A_n = \Box A_m$. If v gives 0 to it, then the set of formulas belonging to the scope of v's necessities which are equivalent to A_m must be empty. This is actually what is required to make *RE* validity-preserving: we would not want to have some $\Box A_q$ getting value 1, and A_q being equivalent to A_m—in which case giving 0 to A_n would mess things up. So $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})]$ must be empty. Similarly, the requirement that $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})]$ should be empty guarantees the validity of *Df*$\Diamond$, because we are then sure there is no $\Diamond\neg A_m$, for instance, such that $v(\Diamond\neg A_m) = 0$—in which case $v(\neg\Diamond\neg A_m)$ would be 1, and it would be bad to have $v(\Box A_m) = 0$. The case with $v(A_n) = 1$, as in the normal modal logics, requires that v has had a good behavior. For possibilities, the picture is analogous.

Maybe it is a surprise for the reader that we are not requiring, in the case where $v(\Box A_m) = 0$, that there is some "accessible" valuation giving 0 to A_m. Actually this condition is the one that guarantees that

RN is validity-preserving. But *RN* doesn't hold in **E**, so it doesn't matter if there is or not another valuation with $v(A_m) = 0$. (Things are different in **EN**, as one can see below.)

Having thus defined α_n–valuations, the rest is standard: a function v from FOR into $\{0,1\}$ is a *valuation* iff for every normal sequence α_n, v is an α_n–valuation. Before going into canonical extensions, normality and so on, I would like to show which kind of modifications are needed in order to get valuations for the other classical modal logics. As in the preceding section, we need only some small changes, so I am only going to repeat each time the most important part of the definition. Thus we have:

- for EN:

A) $A_n = \Box A_m$,

I) (*as in* **E**) ... and there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$;

II) ...

a. (*as in* **E**) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$;

b. (*as in* **E**) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$;

B) $A_n = \Diamond A_m$,

I) (*as in* **E**) ... and there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 1$;

II) ...

a. (*as in* **E**) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$;

b. (*as in* **E**) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$.

What was added here was just a requirement like "and there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$" (or 1), what is, as I said, necessary to guarantee the validity of *RN*. If a necessity is false, then somewhere its scope must get a 0. But this is all: no need that this other α_{n-1}–valuation be in any kind of relation to v.

Let us now see how things look in the case of the other classical logics.

- for EM:

A) $A_n = \Box A_m$,

I) (*as in* **E**)

II) ...

a. (*as in* **E**)

b. $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

B) $A_n = \Diamond A_{m}$,

I) if $v(\mathrm{An}) = 1$ then $\eta^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \zeta^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

II) ...

a. (*as in* **E**)

b. $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}) \cup \{A_m\}] = \emptyset$.

• for EMN:

The only difference in relation to **EM** is that one should add, as in previous cases, the "and there is an α_{n-1}–valuation..." story, which takes care of *RN*.

For the other logics, now, the differences are somewhat greater, so let us write them down whole.

• for EC, EMC:

A) $A_n = \Box A_m$,

I) if $v(A_n) = 0$ then $\xi^{n-1}\{A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

II) if $v(A_n) = 1$ then for every p, every q, $q < p \leq n$, such that:

a. $A_p = \Box A_q$ and $v(A_p) = 0$, $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

b. $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \emptyset$;

B) $A_n = \Diamond A_m$,

I) if $v(\mathrm{An}) = 1$ then $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \emptyset$;

II) if $v(A_n) = 0$ then for every p, every q, $q < p \leq n$, such that:

a. $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \emptyset$;

b. $A_p = \Box A_q$ and $v(A_p) = 0$, $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}) \cup \{A_m\}] = \emptyset$.

It is probably not necessary to say that, even if the definition looks the same for EC and EMC, 'ξ' and 'χ' abbreviate different things! Now for **ECN** and **EMCN** all we need is to add the "and there is an α_{n-1}–valuation..." story, which takes care of *RN* (see the case of **EN**). I hope there is no need to repeat the definition, because I won't. By the way, since **EMCN** is the same logic **K**, we have here an alternative definition of an α_n–valuation for **K**. It is a good exercise for the reader to prove the equivalence of this defintion with the one given in the section on normal modal logics!

The next definition now considers the *canonical extensions* (first for the logic **E**).

Definition D12. Let α_n be a normal sequence and v an α_{n-1}–valuation. We say that v_c is the *canonical extension* of v to α_n if:

A) for ali $m < n$, $A_n \neq \Box A_m$, $A_n \neq \Diamond A_m$ and $v_c = v$; or

B) for some $m < n$, $A_n = \Box A_m$ or $A_n = \Diamond A_m$ and v_c is a function from FOR into $\{0,1\}$ such that, for every formula **B**,

1) if A_n is not a subformula of B, then $v_c(B) = v(B)$;

2) if A_n is a subformula of B, then

a) for $B = A_n = \Box A_m$, $v_c(B) = 0$ iff $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

a') for $B = A_n = \Diamond A_m$, $v_c(B) = 1$ iff $\chi^{n-1}\{A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})\} = \xi^{n-1}\{A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

b) for $B = \neg C$, $v_c(B) = 1$ iff $v_c(C) = 0$;

c) for $B = C \rightarrow D$, $v_c(B) = 1$ iff $v_c(C) = 0$ or $v_c(D) = 1$;

d) for $B = \Box C$ or $B = \Diamond C$, $v_c(B) = v(B)$.

This definition now must undergo some changes, in order to be adequate to the other logics. We have:

- for **EN**:

a) (*as in* **E**) ... and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$;
a') (*as in* **E**) ... and there is an α_{n-1}–valuation v^+ such that $v^+(\mathrm{A_m}) = 1$;

- for **EM**:

a) (*as in* **E**);
a') for $B = A_n = \Diamond A_m$, $v_c(B) = 1$ iff $\eta^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\square}{}_{v,1})] = \zeta^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

- for **EMN**:

a) (*as in* **EM**) ... and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$;
a') (*as in* **EM**) ... and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 1$;

- for **EC, EMC**:

a) for $B = A_n = \square A_m$, $v_c(B) = 0$ iff $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\square}{}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;
a') for $B = A_n = \Diamond A_m$, $v_c(B) = 1$ iff $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}), \varepsilon(\{\alpha_{n-1}\}^{\square}{}_{v,1})] = \emptyset$;

- for **ECN, EMCN**:

a) (*as in* **EC, EMC**) ... and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$;
a') (*as in* **EC, EMC**) ... and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 1$;

In the next step, as usual, we introduce the notion of *normality*. Let v be an α_n–valuation: for $1 \leq k \leq n$, we say that:

- for **E**:

(a) v is $\square$–α_k–*normal* iff for every p, every q, $q < p \leq k$, such that $A_p = \square A_q$ and $v(A_p) = 0$, $\xi^k[A_q, \varepsilon(\{\alpha_k\}^{\square}{}_{v,1})] = \chi^k[A_q, \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0})] = \emptyset$;
(b) v is $\Diamond$–α_k–*normal* iff for every p, every q, $q < p \leq k$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\chi^k[A_q, \varepsilon(\{\alpha_k\}^{\square}{}_{v,1})] = \xi^k[A_q, \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0})] = \emptyset$.

- for **EN**:

(a) (*as in* **E**) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$;
(b) (*as in* **E**) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$.

• for EM:

(a) (*as in* E)
(b) v is $\Diamond$–α_k–*normal* iff for every p, every q, $q < p \le k$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\eta^k[A_q, \varepsilon(\{\alpha_k\}^{\Box}{}_{v,1})] = \zeta^k[A_q, \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0})] = \emptyset$.

• for EMN:

(a) (*as in* EM) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$;
(b) (*as in* EM) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$.

• for EC, EMC:

(a) v is $\Box$–α_k–*normal* iff for every p, every q, $q < p \le k$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, $\xi^k[A_q, \varepsilon(\{\alpha_k\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0})] = \emptyset$;
(b) v is $\Diamond$–α_k–*normal* iff for every p, every q, $q < p \le k$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$, $\chi^k[A_q, \varepsilon(\{\alpha_k\}^{\Diamond}{}_{v,0}), \varepsilon(\{\alpha_k\}^{\Box}{}_{v,1})] = \emptyset$.

• for ECN, EMCN:

(a) (*as in* EC, EMC) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$;
(b) (*as in* EC, EMC) ... and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$.

Now the story repeats itself, just like in normal modal logics. We prove that canonical extensions are semi-valuations, and then that they are α_n-valuations, if normal. Last but not least, we prove that α_n-valuations are normal up to n.

Lemma L15. *If v is an α_{n-1}–valuation and v_c is the canonical extension from v to α_n, then v_c is a semi-valuation.*

Proposition P23. *Let α_n be a normal sequence, v an α_{n-1}–valuation and v_c the canonical extension of v to α_n. Let us suppose that v is $\Box$– and $\Diamond$–α_{n-1}–normal. In this case, v_c is an α_n–valuation.*

Proof First of ali, v_c is an α_{n-1}–valuation, because it is a semi–valuation and, by construction, for $1 \le i < n$, $v_c(A_i) = v(A_i)$. Now, if, for every $m < n$, $A_n \ne \Box A_m$, $A_n \ne \Diamond A_m$, v_c fulfills every condition of Definition D11, so it is an α_n–valuation. Suppose, then, that for some $m < n$, $A_n = \Box A_m$. We have two main cases, and a lot of subcases:

(I) $v_c(A_n) = 0$.

(A) E, EN, EM, EMN:

By D12.B.2.a $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$ [EN, EMN: and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$]. Since v and v_c agree for $i < n$, ξ^{n-1} $[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v_c,1})] = \chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v_c,0})] = \emptyset$. So v_c is an α_n–valuation.

(B) EC, ECN, EMC, EMCN:

By D12.B.2.a $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] = \emptyset$ [ECN, EMCN: and there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$]. Since v and v_c agree for $i < n$, $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] = \emptyset$. So v_c is an α_n–valuation.

(II) $v_c(A_n) = 1$. We consider the cases for the different logics:

(A) E, EN, EM, EMN:

By D12.B.2.a, we have:

(1) either $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] \neq \emptyset$, or $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] \neq \emptyset$, or (in EN, EMN) for every α_{n-1}–valuation v^+, $v^+(A_m) = 1$.

Suppose first there is $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$. Then $v(A_p) = 0$ and, since v is $\Box$–α_{n-1}–normal, $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] = \emptyset$ [EN, EMN: and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$]. Since v and v_c agree for $i < n$, we have:

(2) $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] = \emptyset$.

We have now to prove that $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$. Let us suppose this set is not empty: from (2) it follows that $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \{A_m\}$.

a. In E and EN, we thus have that $A_q \approx_{n-1} A_m$, and then it is easy to see, from (2) again and from the fact that A_q and A_m have the same value, that (i) $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \emptyset$ and that (ii) $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] = \emptyset$. In EN, since $A_q \approx_{n-1} A_m$, and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$, we also have (iii) $v_p(A_m) = 0$. Now (i) and (ii), and in EN (iii) too, contradict (1), so $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$.

b. In EM and EMN, we thus have that $A_m \approx>_{n-1} A_q$. Now let us examine the case where $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] \neq \emptyset$. So there is some $\Box A_j$, $v(\Box A_j) = 1$, such that $A_j \approx>_{n-1} A_m$. It follows, since $A_m \approx>_{n-1} A_q$, that $A_j \approx>_{n-1} A_q$. So $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] \neq \emptyset$, against (2). Well, then $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})]$ must be not empty. So there is some $\Diamond A_j$, $v(\Diamond A_j) = 0$, such that $A_j <\approx_{n-1} A_m$. It follows that $A_j <\approx_{n-1} A_q$, and then $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] \neq \emptyset$, again against (2). Now suppose, in EMN, that for every α_{n-1}–valuation v^+, $v^+(A_m) = 1$. Since $A_m \approx>_{n-1} A_q$, we have that for every α_{n-1}–valuation v^+, $v^+(A_q) = 1$. So $v_p(A_q) = 1$, what cannot be. Hence, in both logics, $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$.

Suppose now there is $q < p \leq n$ such that $A_p = \Diamond A_q$ and $v_c(A_p) = 1$. Then $v(A_p) = 1$ and, since v is $\Diamond$–α_{n-1}–normal, we have: $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] = \emptyset$ [EN, EMN: and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$]. Since v and v_c agree for $i < n$, we have:

a. In E and EN:

(3) $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] = \emptyset$ [EN: and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$].

Since v and v_c agree for $i < n$, we have:

(4) $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] = \emptyset$.

We have now to prove that $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$. Suppose it is not: it follows that $A_q \approx_{n-1} A_m$. Now, if $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})]$ were not empty, there would be an $A_i \in \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})$

such that $A_i \approx_{n-1} A_m$. So $A_q \not\approx_{n-1} A_i$, and then $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] \neq \emptyset$, against (4). So (i) $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \emptyset$. Then $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})]$ must be not empty; so there is an $A_i \in \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})$ such that $A_i \not\approx_{n-1} A_m$. It follows that $A_q \approx_{n-1} A_i$. But then, against (4), $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})] \neq \emptyset$, so (ii) $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})] = \emptyset$. Now, in **EN**, since $A_q \not\approx_{n-1} A_m$, and since there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$, we also have (iii) $v_p(A_m) = 0$. But (i) and (ii), and in **EN** (iii) too, contradict (1), so $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$.

b. In **EM** and **EMN**:

(5) $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})] = \emptyset$ [**EMN**: and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$].

Since v and v_c agree for $i < n$, we have:

(6) $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] = \zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})] = \emptyset$.

We have to prove that $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] \neq \emptyset$. Suppose it is not: it follows that $A_q \not\approx\!>_{n-1} A_m$. Now, if $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})]$ were not empty, there would be an $A_i \in \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})$ such that $A_i \approx\!>_{n-1} A_m$. It is easy to see that $A_q \not\approx\!>_{n-1} A_i$, and then $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] \neq \emptyset$, against (6). So (i) $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \emptyset$. Then $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})]$ must be not empty. Then there is an $A_i \in \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})$ such that $A_i <\!\not\approx_{n-1} A_m$. It follows easily that $A_q \approx\!>_{n-1} A_i$. But then, against (6), $\zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})] \neq \emptyset$, so (ii) $\chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})] = \emptyset$. Suppose now, in **EMN**, that for every α_{n-1}–valuation v^+, $v^+(A_m) = 1$. Since $A_q \not\approx\!>_{n-1} A_m$, we have that for every α_{n-1}–valuation v^+, $v^+(A_q) = 1$. So $v_p(A_q) = 1$, what cannot be. So (iii) there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$. However, (i) and (ii), and in **EMN** (iii) too, contradict (1), so $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$.

It follows, in case (A), that v_c is an α_n–valuation. Let us consider the other logics:

(B) **EC**, **ECN**, **EMC**, **EMCN**:

By D12.B.2.a, we have:

(1) $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})] \neq \emptyset$, or, in **ECN** and **EMCN**, for every α_{n-1}–valuation v^+, $v^+(A_m) = 1$].

Suppose first there is $q < p \leq n$ such that $A_p = \Box A_q$ and $v_c(A_p) = 0$. Then $v(A_p) = 0$ and, since v is $\Box$–α_{n-1}–normal, $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{v,0})] = \emptyset$ [**ECN**, **EMCN**: and there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$]. Since v and v_c agree for $i < n$, we have:

(2) $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}), \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})] = \emptyset$.

We have now to prove that $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})] = \emptyset$. Let us suppose this set is not empty: from (2) it follows that, for some $\Gamma \subseteq \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})$, some $\Delta \subseteq \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})$, $<\Gamma \cup \{A_m\}, \Delta> \in \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\diamond}_{vc,0})]$. By definition, for every α_{n-1}–valuation v':

(i^a) [**EC**, **ECN**] $v'(A_q) = 1$ iff $v' \vDash_1 \Gamma \cup \{A_m\}$ and $v' \vDash_0 \Delta$, i.e.: $v'(A_q) = 1$ iff $v' \vDash_1 \Gamma$ and $v' \vDash_0 \Delta$ and $v'(A_m) = 1$.

(i^b) [**EMC**, **EMCN**] if $v' \vDash_1 \Gamma \cup \{A_m\}$ and $v' \vDash_0 \Delta$ then $v'(A_q) = 1$, i.e.: if $v' \vDash_1 \Gamma$ and $v' \vDash_0 \Delta$ and $v'(A_m) = 1$ then $v'(A_q) = 1$.

Now, suppose, in **ECN** and **EMCN**, that $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] \neq \emptyset$. (In **EC** and **EMC** it is already so.) It follows that, for some $\Theta \subseteq \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})$, some $\Phi \subseteq \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})$, $<\Theta,\Phi> \in \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})]$. By definition, we have, for every α_{n-1}–valuation v':

(ii^a) [**EC, ECN**] $v'(A_m) = 1$ iff $v' \vDash_1 \Theta$ and $v' \vDash_0 \Phi$.

(ii^b) [**EMC, EMCN**] if $v' \vDash_1 \Theta$ and $v' \vDash_0 \Phi$ then $v'(A_m) = 1$.

From (i-ii^a) and (i-ii^b) we get, for every α_{n-1}–valuation v':

(iii^a) [**EC, ECN**] $v'(A_q) = 1$ iff $v' \vDash_1 \Gamma$ and $v' \vDash_0 \Delta$ and $v' \vDash_1 \Theta$ and $v' \vDash_0 \Phi$.

(iii^b) [**EMC, EMCN**] if $v' \vDash_1 \Gamma$ and $v' \vDash_0 \Delta$ and $v' \vDash_1 \Theta$ and $v' \vDash_0 \Phi$ then $v'(A_q) = 1$.

But this, for every α_{n-1}–valuation v', is the same as:

(iv^a) [**EC, ECN**] $v'(A_q) = 1$ iff $v' \vDash_1 \Gamma \cup \Theta$ and $v' \vDash_0 \Delta \cup \Phi$.

(iv^b) [**EMC, EMCN**] if $v' \vDash_1 \Gamma \cup \Theta$ and $v' \vDash_0 \Delta \cup \Phi$ then $v'(A_q) = 1$.

But then there is a pair $<\Gamma \cup \Theta, \Delta \cup \Phi> \in \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})]$, what cannot be. Thus:

(v) $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] = \emptyset$.

In **ECN** and **EMCN** we still have to consider the other half of the disjunction in (1). So suppose that for every α_{n-1}–valuation v', $v'(A_m) = 1$. From (i$^{a\text{-}b}$) we have, for every α_{n-1}–valuation v':

(vi^a) [**ECN**] $v'(A_q) = 1$ iff $v' \vDash_1 \Gamma$ and $v' \vDash_0 \Delta$.

(vi^b) [**EMCN**] if $v' \vDash_1 \Gamma$ and $v' \vDash_0 \Delta$ then $v'(A_q) = 1$.

But then there is a pair $<\Gamma, \Delta> \in \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})]$, what cannot be. Thus also here we have that

(vii) $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{\mathrm{n-1}}\}^{\Box}_{vc,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})] = \emptyset$.

Suppose now there is $q < p \leq n$ such that $A_p = \Diamond A_q$ and $v_c(A_p) = 1$. Then $v(A_p) = 1$ and, since v is $\Diamond$–α_{n-1}–normal, $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1})] = \emptyset$. Since v and v_c agree for $i < n$, we have:

(3) $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})] = \emptyset$.

We have now to prove that $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}] = \emptyset$. Suppose it is not: it follows that, for some $\Gamma \subseteq \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})$, for some $\Delta \subseteq \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1})$, $<\Gamma, \Delta \cup \{A_m\}> \in \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}) \cup \{A_m\}]$. By definition, for every α_{n-1}–valuation v':

(i^a) [**EC, ECN**] $v'(A_q) = 0$ iff $v' \vDash_0 \Gamma$ and $v' \vDash_1 \Delta \cup \{A_m\}$, i.e.: $v'(A_q) = 0$ iff $v' \vDash_0 \Gamma$ and $v' \vDash_1 \Delta$ and $v'(A_m) = 1$.

(i^b) [**EMC, EMCN**] if $v' \vDash_0 \Gamma$ and $v' \vDash_1 \Delta \cup \{A_m\}$ then $v'(A_q) = 0$, i.e.: if $v' \vDash_0 \Gamma$ and $v' \vDash_1 \Delta$ and $v'(A_m) = 1$ then $v'(A_q) = 0$.

Now suppose, in **ECN** and **EMCN**, that $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0})] \neq \emptyset$. (In **EC** and **EMC** it is already so.) It follows that, for some $\Theta \subseteq \varepsilon(\{\alpha_{\mathrm{n-1}}\}^{\Box}_{vc,1})$, some $\Phi \subseteq \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})$, $<\Theta,\Phi> \in \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}_{vc,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{vc,0})]$. By definition, we have, for every α_{n-1}–valuation v':

(ii^a) [EC, ECN] $v'(A_m) = 1$ iff $v' \models_1 \Theta$ and $v' \models_0 \Phi$.

(ii^b) [EMC, EMCN] if $v' \models_1 \Theta$ and $v' \models_0 \Phi$ then $v'(A_m) = 1$.

From (i$^{a\text{-}b}$) and (ii$^{a\text{-}b}$), then, for every $\alpha_{n\text{-}1}$–valuation v':

(iii^a) [EC, ECN] $v'(A_q) = 0$ iff $v' \models_0 \Gamma$ and $v' \models_1 \Delta$ and $v' \models_1 \Theta$ and $v' \models_0 \Phi$.

(iii^b) [EMC, EMCN] if $v' \models_0 \Gamma$ and $v' \models_1 \Delta$ and $v' \models_1 \Theta$ and $v' \models_0 \Phi$ then $v'(A_q) = 0$.

That is, for every $\alpha_{n\text{-}1}$–valuation v',

(iv^a) [EC, ECN] $v'(A_q) = 0$ iff $v' \models_0 \Gamma \cup \Phi$ and $v' \models_1 \Delta \cup \Theta$.

(iv^b) [EMC, EMCN] if $v' \models_0 \Gamma \cup \Phi$ and $v' \models_1 \Delta \cup \Theta$ then $v'(A_q) = 0$.

But then there is a pair $<\Gamma \cup \Theta, \Delta \cup \Phi> \in \chi^{n\text{-}1}[A_q, \varepsilon(\{\alpha_{n\text{-}1}\}^{\diamond}{}_{vc,0}), \varepsilon(\{\alpha_{n\text{-}1}\}^{\Box}{}_{vc,1})]$, what cannot be. Thus

(v) $\chi^{n\text{-}1}[A_q, \varepsilon(\{\alpha_{n\text{-}1}\}^{\diamond}{}_{vc,0}), \varepsilon(\{\alpha_{n\text{-}1}\}^{\Box}{}_{vc,1}) \cup \{A_m\}] = \emptyset$.

In ECN and EMCN we still have to consider the other half of the disjunction in (1). So suppose that for every $\alpha_{n\text{-}1}$–valuation v', $v'(A_m) = 1$. From (i$^{a\text{-}b}$) we have, for every $\alpha_{n\text{-}1}$–valuation v',

(vi^a) [ECN] $v'(A_q) = 0$ iff $v' \models_0 \Gamma$ and $v' \models_1 \Delta$.

(vi^b) [EMCN] if $v' \models_0 \Gamma$ and $v' \models_1 \Delta$ then $v'(A_q) = 0$.

But then there is a pair $<\Gamma, \Delta> \in \chi^{n\text{-}1}[A_q, \varepsilon(\{\alpha_{n\text{-}1}\}^{\diamond}{}_{vc,0}), \varepsilon(\{\alpha_{n\text{-}1}\}^{\Box}{}_{vc,1})]$, what cannot be. Thus also here we have that

(vii) $\chi^{n\text{-}1}[A_q, \varepsilon(\{\alpha_{n\text{-}1}\}^{\diamond}{}_{vc,0}), \varepsilon(\{\alpha_{n\text{-}1}\}^{\Box}{}_{vc,1}) \cup \{A_m\}] = \emptyset$.

It follows, also in case (B), that v_c is an α_n–valuation.

The proof for $A_n = \diamond A_m$ is analogous.■

We have thus proved that canonical extensions are α_n–valuations on the hypothesis that the $\alpha_{n\text{-}1}$–valuations they are extending are normal. With the next lemma, we can show that α_n–valuations are normal without restrictions.

Lemma L16. (*Normality Lemma*) *Let v be an α_n–valuation. Then v is* $\Box$– *and* $\diamond$–α_n*–normal.*

Proof. By induction on n. For $n = 1$ it holds trivially, so let $n > 1$ and let us suppose that every $\alpha_{n\text{-}1}$–valuation is $\Box$– and $\diamond$–$\alpha_{n\text{-}1}$–normal. It follows then from P23 that

(†) The canonical extensions of $\alpha_{n\text{-}1}$–valuations to α_n are α_n–valuations.

Moreover, it is trivially true that, for $i < n$, and $\Gamma \subseteq \{\alpha_{n\text{-}1}\}$,

(1) $\xi^{n\text{-}1}[A_i, \Gamma] = \xi^n[A_i, \Gamma]$;

(2) $\chi^{n\text{-}1}[A_i, \Gamma] = \chi^n[A_i, \Gamma]$;

(3) $\zeta^{n\text{-}1}[A_i, \Gamma] = \zeta^n[A_i, \Gamma]$;

(4) $\eta^{n-1}[A_i, \Gamma] = \eta^n[A_i, \Gamma]$;

(5) $\xi^{n-1}[A_i, \Gamma, \Delta] = \xi^n[A_i, \Gamma, \Delta]$;

(6) $\chi^{n-1}[A_i, \Gamma, \Delta] = \chi^n[A_i, \Gamma, \Delta]$;

because every α_n–valuation is an α_{n-1}–valuation.

We have now three main cases:

(A) For every $m < n$, $A_n \neq \Box A_m$, $A_n \neq \Diamond A_m$. So v is trivially $\Box$– and $\Diamond$–α_n–normal.

(B) Let us suppose that, for some $m < n$, $A_n = \Box A_m$.

(I) Let $v(A_n) = 0$. We have:

(7) $\varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}) = \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})$;

(8) $\varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0}) = \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})$.

From the induction hypothesis, v is $\Box$–α_{n-1}–normal, so we have:

(9) for every p, every q, $q < p < n$ such that $A_p = \Box A_q$ and $v(A_p) = 0$,

(9ᵃ) **E, EN, EM, EMN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

(9ᵇ) **EC, ECN, EMC, EMCN**: $\xi^{n-1}[\mathrm{A_q}, \varepsilon(\{\mathrm{a_{n-1}}\}^{\Box}{}_{v,1}), \varepsilon(\{\mathrm{a_{n-1}}\}^{\Diamond}{}_{v,0})] = \emptyset$;

moreover, in **EN, EMN, ECN** and **EMCN**, there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$.

From (1), (2), (7) and (8), we get:

(10ᵃ) **E, EN, EM, EMN**: $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$.

(10ᵇ) **EC, ECN, EMC, EMCN**: $\xi^n[\mathrm{A_q}, \varepsilon(\{\mathrm{a_n}\}^{\Box}{}_{v,1}), \varepsilon(\{\mathrm{a_n}\}^{\Diamond}{}_{v,0})] = \emptyset$.

In **EN, EMN, ECN** and **EMCN**, for each p, let $v_p{}^*$ be the canonical extension from v_p to α_n. Obviously $v_p{}^*(A_q) = v_p(A_q)$, and, from (†), $v_p{}^*$ is an α_n–valuation. From this, (9) and (10), then:

(11) for every p, every q, $q < p < n$ sucht that $A_p = \Box A_q$ and $v(A_p) = 0$,

(11ᵃ) **E, EN, EM, EMN**: $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$;

(11ᵇ) **EC, ECN, EMC, EMCN**: $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})], \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$;

moreover, in **EN, EMN, ECN** and **EMCN**, there is an α_n–valuation $v_p{}^*$ such that $v_p{}^*(A_q) = 0$.

On the other hand, since v is an α_n–valuation, we have:

(12ᵃ) **E, EN, EM, EMN**: $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$ [**EN, EMN**: and there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$].

(12ᵇ) **EC, ECN, EMC, EMCN**: $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$ [**ECN, EMCN**: and there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$].

From (1), (2), (7), (8), and from the fact that $A_m \in \{\alpha_{n-1}\}$, we get:

(13ᵃ) **E, EN, EM, EMN**: $\xi^n[A_m, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \chi^n[A_m, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$.

(13ᵇ) **EC, ECN, EMC, EMCN**: $\xi^n[A_m, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$.

In **EN**, **EMN**, **ECN** and **EMCN**, let $v_n{}^*$ be the canonical extension from v_n to α_n. Obviously $v_n{}^*(A_m) = v_n(A_m)$, and, from (†), $v_n{}^*$ is an α_n–valuation. Thus we have

(14) for $p = n$, $q = m$, $A_p = \Box A_q$ and $v(A_p) = 0$,

(14^a) **E**, **EN**, **EM**, **EMN**: $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$;

(14^b) **EC**, **ECN**, **EMC**, **EMCN**: $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$;

moreover, in **EN**, **EMN**, **ECN** and **EMCN**, there is an α_n–valuation $v_p{}^*$ such that $v_p{}^*(A_q) = 0]$.

From this, together with (11), then, v is an $\Box$–α_n–normal.

Now, from the induction hypothesis, v is $\Diamond$–α_{n-1}–normal, so we have, together with the fact that $A_n = \Box A_m$:

(15) for every p, every q, $q < p < n$ such that $A_p = \Diamond A_q$ and $v(A_p) = 1$,

(15^a) **E**, **EN**: $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

(15^b) **EM**, **EMN**: $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

(15^c) **EC**, **ECN**, **EMC**, **EMCN**: $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1})] = \emptyset$;

moreover, in **EN**, **EMN**, **ECN** and **EMCN**, there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$.

From (1), (2), (7) and (8), we get:

(16^a) **E**, **EN**: $\eta^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \zeta^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$;

(16^b) **EM**, **EMN**: $\chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})] = \emptyset$;

(16^c) **EC**, **ECN**, **EMC**, **EMCN**: $\chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0}), \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})] = \emptyset$.

In **EN**, **EMN**, **ECN** and **EMCN**, for each p, let $v_p{}^*$ be the canonical extension from v_p to α_n. Obviously $v_p{}^*(A_q) = v_p(A_q)$, and, from (†), $v_p{}^*$ is an α_n–valuation. From this and (16^{a-c}), then, v is $\Diamond$–α_n–normal.

(II) Let $v(A_n) = 1$. We then have:

(7) $\varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}) = \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}$;

(8) $\varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0}) = \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})$.

Since $v(A_n) = 1$, we have from D11 that:

(9) for every p, every q, $q < p < n$ such that:

i. $A_p = \Box A_q$ and $v(A_p) = 0$,

(^{a}) **E**, **EN**, **EM**, **EMN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

(^{b}) **EC**, **ECN**, **EMC**, **EMCN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

moreover, in **EN**, **EMN**, **ECN** and **EMCN**, there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$;

ii. $A_p = \Diamond A_q$ and $v(A_p) = 1$,

(^{a}) **E**, **EN**: $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

(^{b}) **EM**, **EMN**: $\eta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Box}{}_{v,1}) \cup \{A_m\}] = \zeta^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}{}_{v,0})] = \emptyset$;

(c) **EC, ECN, EMC, EMCN:** $\chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_{n-1}\}^{\Box}_{v,1}) \cup \{A_m\}] = \emptyset$;

moreover, in **EN, EMN, ECN** and **EMCN**, there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 1$.

In **EN, EMN, ECN** and **EMCN**, for each p, let v_p* be the canonical extension from v_p to α_n. Obviously $v_p{}^*(A_q) = v_p(A_q)$, and, from (†), v_p* is an α_n–valuation. From (1), (2), (7), (8) and (9), and from the fact that $A_m \in \{\alpha_{n-1}\}$, we get:

(10) for every p, every q, $q < p \leq n$ such that:

i. $A_p = \Box A_q$ and $v(A_p) = 0$,

(a) **E, EN, EM, EMN:** $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}_{v,1}) \cup \{A_m\}] = \chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})] = \emptyset$;

(b) **EC, ECN, EMC, EMCN:** $\xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}_{v,1}) \cup \{A_m\}, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})] = \emptyset$;

moreover, in **EN, EMN, ECN** and **EMCN**, there is an α_n–valuation v_p such that $v_p(A_q) = 0$;

ii. $A_p = \Diamond A_q$ and $v(A_p) = 1$,

(a) **E, EN:** $\chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}_{v,1}) \cup \{A_m\}] = \xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})] = \emptyset$;

(b) **EM, EMN:** $\eta^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}_{v,1}) \cup \{A_m\}] = \zeta^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})] = \emptyset$;

(c) **EC, ECN, EMC, EMCN:** $\chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_n\}^{\Box}_{v,1}) \cup \{A_m\}] = \emptyset$;

moreover, in **EN, EMN, ECN** and **EMCN**, there is an α_n–valuation v_p such that $v_p(A_q) = 1$.

That is, v is $\Box$– and $\Diamond$–α_n–normal.

(C) Let us suppose that, for some $m < n$, $A_n = \Diamond A_m$. Proof as in (B).■

As a direct result of combining this Lemma with P23, we have the following

Corollary. *Let α_n be a normal sequence, v an α_{n-1}–valuation and v_c the canonical extension of v to α_n. Then v_c is an α_n–valuation and, for $1 \leq i \leq n-1$, $v_c(A_i) = v(A_i)$.*

The next theorem makes use of all we got until now:

Theorem T23. *v is an α_n–valuation iff: 1) α_n is a normal sequence; 2) v is a semi-valuation; 3) v is $\Box$– and $\Diamond$–α_n–normal.*

5.2 Correctness

Having then proved these properties of (α_n–)valuations, we are now ready to consider correctness. The strategy is analogous to the case of normal logics.

Lemma L17. *Let v be an α_n-valuation; then, for $1 \leq i \leq n$, if A_i is an axiom of some classical modal logic L then $v(A_i) = 1$.*

Proof. The axioms of said logics are either those from **PL**, and it follows from the fact that v is a semi-valuation, that $v(A_i) = 1$, or they are one of the modal axiom schemes. We consider each case.

(a) $A_i = \Diamond A \leftrightarrow \neg\Box\neg A$. Suppose $v(\Diamond A \leftrightarrow \neg\Box\neg A) = 0$. Then we have, say, $v(\Diamond A) = 1$ and $v(\neg\Box\neg A) = 0$, so $v(\Box\neg A) = 1$. From the normality lemma it follows that, for every p, every q, $q < p \leq n$, such that $A_p = \Diamond A_q$ and $v(A_p) = 1$:

(i) E, EN:

$\chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}_{v,1})] = \xi^n[A_q, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})] = \emptyset$. But, for every semi-valuation, and, consequently, for every α_n-valuation, $A \not\approx_n \neg A$, so $\neg A \in \chi^n[A_q, \varepsilon(\{\alpha_n\}^{\Box}_{v,1})]$, what cannot be.

(ii) EM, EMN:

$\eta^n[A, \varepsilon(\{\alpha_n\}^{\Box}_{v,1})] = \zeta^n[A, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})] = \emptyset$. But, for every semi-valuation, and, consequently, for every α_n-valuation, $A \not\approx>_n \neg A$; hence, since $\neg A \in \varepsilon(\{\alpha_n\}^{\Box}_{v,1})$, $\neg A \in \eta^n[A, \varepsilon(\{\alpha_n\}^{\Box}_{v,1})]$, what cannot be.

(iii) EC, ECN:

$\chi^n[A, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_n\}^{\Box}_{v,1})] = \emptyset$. Let us take the pair $<\emptyset, \{\neg A\}>$. Obviously $\emptyset \subseteq \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})$, $\{\neg A\} \subseteq \varepsilon(\{\alpha_n\}^{\Box}_{v,1})$ and for every α_n-valuation v', $v'(A) = 0$ iff $v' \models_0 \emptyset$ and $v' \models_1 \{\neg A\}$; that is, $A \not\approx_n <\emptyset, \{\neg A\}>$. Hence $<\emptyset, \{\neg A\}> \in \chi^n[A, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_n\}^{\Box}_{v,1})]$, what cannot be.

(iv) EMC, EMCN:

$\chi^n[A, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_n\}^{\Box}_{v,1})] = \emptyset$. Let us take the pair $<\emptyset, \{\neg A\}>$. Obviously $\emptyset \subseteq \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})$, $\{\neg A\} \subseteq \varepsilon(\{\alpha_n\}^{\Box}_{v,1})$ and for every α_n-valuation v', if $v' \models_0 \emptyset$ and $v' \models_1 \{\neg A\}$ then $v'(A) = 0$; that is, $<\emptyset, \{\neg A\}> \not\approx>_n A$. Hence $<\emptyset, \{\neg A\}> \in \chi^n[A, \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0}), \varepsilon(\{\alpha_n\}^{\Box}_{v,1})]$, what cannot be.

If now $v(\Diamond A) = 0$ and $v(\neg\Box\neg A) = 1$, the proof goes for every logic in a similar way. Hence $v(\Diamond A \leftrightarrow \neg\Box\neg A) = 1$.

We must now consider the special axioms of each system.

(b) $A_i = \Box A \wedge \Box B \rightarrow \Box(A \wedge B)$ (EC, EMC, ECN, EMCN).

Suppose $v(A_i) = 0$. So $v(\Box(A \wedge B)) = 0$ and $v(\Box A) = v(\Box B) = 1$. Let us take the pair $<\{A, B\}, \emptyset>$. Obviously $\{A,B\} \subseteq \varepsilon(\{\alpha_n\}^{\Box}_{v,1})$, $\emptyset \subseteq \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})$, and for every α_n-valuation v',

[EC, ECN] $v'(A \wedge B) = 1$ iff $v' \models_1 \{A,B\}$ and $v' \models_0 \emptyset$; that is, $A \approx_n <\{A,B\}, \emptyset>$.

[EMC, EMCN] if $v' \models_1 \{A,B\}$ and $v' \models_0 \emptyset$ then $v'(A \wedge B) = 1$; that is, $<\{A,B\}, \emptyset> \approx>_n A$.

Hence $<\{A,B\}, \emptyset> \in \xi^n[A, \varepsilon(\{\alpha_n\}^{\Box}_{v,1}), \varepsilon(\{\alpha_n\}^{\Diamond}_{v,0})]$, what cannot be. So $v(A_i) = 1$.

(c) $A_i = \Box(A \wedge B) \rightarrow \Box A \wedge \Box B$ (EM, EMC, EMN, EMCN).

Suppose $v(A_i) = 0$. So $v(\Box(A \wedge B)) = 1$ and $v(\Box A) = 0$, or $v(\Box B) = 0$. We consider the logics separately:

[EM, EMN] Obviously $A \wedge B \approx>_n A$, and $A \wedge B \approx>_n B$. Since $A \wedge B \in \varepsilon(\{\alpha_n\}^{\Box}_{v,1})$, we have that $\xi^n[A, \varepsilon(\{\alpha_n\}^{\Box}_{v,1})] \neq \emptyset$, and $\xi^n[B, \varepsilon(\{\alpha_n\}^{\Box}_{v,1})] \neq \emptyset$, so $v(\Box A) = v(\Box B) = 1$, what cannot be. So $v(A_i) = 1$.

{EMC, EMCN} Let us take the pair $<\{A \wedge B\}, \emptyset>$. Obviously $\{A \wedge B\} \subseteq \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1})$, $\emptyset \subseteq \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})$, and for every α_n–valuation v', if $v' \models_1 \{A \wedge B\}$ and $v' \models_0 \emptyset$ then $v'(A) = v'(B) = 1$; that is, $<\{A,B\}, \emptyset> \approx>_n A$; $<\{A,B\}, \emptyset> \approx>_n B$. Hence $<\{A \wedge B\}, \emptyset> \in \xi^n[A, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})]$, $<\{A \wedge B\}, \emptyset> \in \xi^n[B, \varepsilon(\{\alpha_n\}^{\Box}{}_{v,1}), \varepsilon(\{\alpha_n\}^{\Diamond}{}_{v,0})]$, what cannot be. So $v(A_i) = 1$.■

Theorem T24. *If A is an axiom of L and v is a valuation, then $v(A) = 1$.*

Proof. Let A be an axiom of one of the said logics, and v a valuation. Let α_n be a normal sequence such that, for some $i \leq n$, $A = A_i$. By definition, v is an α_n–valuation and, from L12, $v(A) = 1$.■

Lemma L18. *For all n, all i, $1 \leq i \leq n$, and L a classical modal logic, if v is an α_n–valuation and $\vdash_L A_i$, then $v(A_i) = 1$.*

Proof. By induction on the number r of lines of a proof of A_i in L.

A) $r = 1$. Then A_i is an axiom, and the property follows from L12.

B) $r > 1$. If A_i is an axiom, the property follows from L12; else:

(a) A_i was obtained by *MP* from B and $B \rightarrow A_i$. Proof as in the case of normal logics, since valuations here also are semi-valuations.

(b) $A_i = \Box B \leftrightarrow \Box C$ and was obtained by *RE* from $B \leftrightarrow C$. We have that $\vdash B \leftrightarrow C$. Obviously, for every normal sequence to which $\Box B \leftrightarrow \Box C$ belongs, B, C, $\Box B$ and $\Box C$ belong too, so they belong to α_n. If $B \leftrightarrow C$ also occurs in α_n, let $\sigma = \alpha_n$ and $v_c = v$. Else let $\sigma = A_1,\ldots,A_n,A_{n+1}$, where $A_{n+1} = B \leftrightarrow C$. σ is obviously a normal sequence, so let v_c be the canonical extension of v to σ. v_c is thus a σ–valuation. Since $\vdash B \leftrightarrow C$, we have by the induction hypothesis that for ali n, if v is an α_n–valuation and $\vdash B \leftrightarrow C$, then $v(B \leftrightarrow C) = 1$. So $v_c(B \leftrightarrow C) = 1$; besides, $B \approx_{n+1} C$. If now $v_c(\Box B \leftrightarrow \Box C)$ were to be 0, we would have, say, $v(\Box B) = 1$ and $v(\Box C) = 0$. From the normality lemma, $\xi^{n+1}[C, \varepsilon(\{\alpha_{n+1}\}^{\Box}{}_{v_c,1})]$ should now be empty, but it isn't, because $B \approx_{n+1} C$ and thus B belongs to it. Thus $v_c(\Box B \leftrightarrow \Box C) = 1$, and hence $v(\Box B \leftrightarrow \Box C) = 1$.

(c) $A_i = \Box B$ and was obtained by *RN* from B (for **EN**, **ECN**, **EMN**, **EMCN**). Well, in every normal sequefice σ in which A_i occurs, B occurs too; so, by the induction hypothesis, for every σ–valuation v, $v(B) = 1$. If now $v(A_i)$ were to be 0, there should be an α_n–valuation v_n such that $v_n(B) = 0$, what cannot be. So $v(A_i) = 1$. ■

Corollary. *If $\vdash A$ then $\models A$.*

Proof. Suppose $\vdash A$, and let v be a valuation. Let α_n be a normal sequence in which, for some $i \leq n$, $A = A_i$. By definition, v is an α_n–valuation, so $v(A) = 1$ from L18, and thus $\models A$. ■

Theorem T25. (*Correctness Theorem*) *If $\Gamma \vdash A$ then $\Gamma \models A$.*

Proof. Suppose $\Gamma \vdash A$, and let $D_1,\ldots,D_r$ be a deduction of A from Γ. We prove the theorem by induction on r.

A) $r = 1$. Then, either $A \in \Gamma$, and we have nothing to prove, or A is an axiom, so A is valid (by the corollary to L18) and $\Gamma \models A$.

B) $r > 1$. If $A \notin \Gamma$ and A is not an axiom, then:
(a) for some $j < r$, $i < r$, $D_i = D_j \rightarrow A$. So $\Gamma \vdash D_i$, $\Gamma \vdash D_i \rightarrow A$ and, by the induction hypothesis, $\Gamma \models D_i$, $\Gamma \models D_i \rightarrow A$. Thus, for every valuation v, if $v \models \Gamma$, $v(D_j) = v(D_j \rightarrow A) = 1$, and hence $v(A) = 1$. So $\Gamma \models A$.
(b) $A = \Box B \leftrightarrow \Box C$ and, for some $j < r$, $D_r = B \leftrightarrow C$. In this case, $\vdash D_r$ and $\vdash A$, so, for every valuation v, $v(A) = 1$. Hence $\Gamma \models A$.
(c) $A = \Box B$ and, for some $j < r$, $D_r = B$ (for **EN**, **ECN**, **EMN**, **EMCN**). In this case, $\vdash D_r$ and $\vdash A$. By the Corollary to L18, for every valuation v, $v(A) = 1$. So, if $v \models \Gamma$, $v(A) = 1$. Thus $\Gamma \models A$. ■

5.3 Completeness

Completeness will now be easily proved in the same way of normal logics—that is, making use of saturated sets. In the following, let L be a classical modal logic, and let us understand '$\vdash$' as referring to L. We are now able to prove some results about saturated sets, showing some of their properties which will be of use in completeness proofs.

Proposition P24. *If Δ is saturated, then:*

(a) $A \in \Delta$ *iff* $\Delta \vdash A$;
(b) $\neg A \in \Delta$ *iff* $A \notin \Delta$;
(c) $A \rightarrow B \in \Delta$ *iff* $A \notin \Delta$ *or* $B \in \Delta$;
(d) $A \leftrightarrow B \in \Delta$ *iff* $A \in \Delta$ and $B \in \Delta$; *or* $A \notin \Delta$ *and* $B \notin \Delta$.

Proposition P25. *If $\Gamma \nvdash A$, then there is an A–saturated set Δ such that $\Gamma \subseteq \Delta$.*

Definition D13. Let Γ, Γ' be any sets, A and B wffs, and Θ, Φ finite sets of wffs:

$\Gamma \subseteq_0 \Gamma'$ $=_{df}$ for every $A \in \Gamma$, $A \notin \Gamma'$;
$\Gamma^{\diamond 0}$ $=_{df}$ $\{\diamond A \notin \Gamma\}$;
$\varepsilon(\Gamma^{\diamond 0})$ $=_{df}$ $\{A : \diamond A \notin \Gamma\}$;
$A \approx_S B$ iff for every saturated set Δ, $A \in \Delta$ iff $B \in \Delta$;
$A \not\approx_S B$ iff for every saturated set Δ, $A \in \Delta$ iff $B \notin \Delta$;
$A \approx>_S B$ iff for every saturated set Δ, if if $A \in \Delta$ then $B \in \Delta$;
$A \not\approx>_S B$ iff for every saturated set Δ, if if $A \in \Delta$ then $B \notin \Delta$;
$A <\not\approx_S B$ iff for every saturated set Δ, if if $A \notin \Delta$ then $B \in \Delta$;
$A \approx_S <\Theta, \Phi>$ iff for every saturated set Δ, $A \in \Delta$ iff $\Theta \subseteq \Delta$ and $\Phi \subseteq_0 \Delta$;
$A \not\approx_S <\Theta, \Phi>$ iff for every saturated set Δ, $A \notin \Delta$ iff $\Theta \subseteq_0 \Delta$ and $\Phi \subseteq \Delta$;
$<\Theta, \Phi> \approx>_S A$ iff for every saturated set Δ, if $\Theta \subseteq \Delta$ and $\Phi \subseteq_0 \Delta$ then $A \in \Delta$;
$<\Theta, \Phi> \not\approx>_S A$ iff for every saturated set Δ, if $\Theta \subseteq_0 \Delta$ and $\Phi \subseteq \Delta$ then $A \notin \Delta$;

Definition D14. Let Γ, Δ be any set of wffs. We define:

(a) for **E**, **EN**:

$\xi[A, \Gamma] =_{df} \quad \{ B \in \Gamma : B \approx_S A \};$

$\chi[A, \Gamma] =_{df} \quad \{ B \in \Gamma : B \not\approx_S A \}.$

(b) for **EC**, **ECN**:

$\xi[A, \Gamma, \Delta] =_{df} \{ <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } A \approx_S <\Theta, \Phi> \};$

$\chi[A, \Gamma, \Delta] =_{df} \{ <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } A \not\approx_S <\Theta, \Phi> \};$

(c) for **EMC**, **EMCN**:

$\xi[A, \Gamma, \Delta] =_{df} \{ <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } <\Theta, \Phi> \approx>_S A \};$

$\chi[A, \Gamma, \Delta] =_{df} \{ <\Theta, \Phi> : \Theta \subseteq \Gamma, \Phi \subseteq \Delta \text{ and } <\Theta, \Phi> \not\approx>_S A \};$

(d) for **EM**, **EMN**:

$\xi[A, \Gamma] =_{df} \quad \{ B \in \Gamma : B \approx>_S A \};$

$\chi[A, \Gamma] =_{df} \quad \{ B \in \Gamma : B <\not\approx_S A \};$

$\zeta[A, \Gamma] =_{df} \quad \{ B \in \Gamma : A \approx>_S B \};$

$\eta[A, \Gamma] =_{df} \quad \{ B \in \Gamma : A \not\approx>_S B \}.$

Lemma L19. *Let* Δ *be a saturated set.*

If $\Delta \nvdash \Box A$, *then:*

(i) **E, EN, EM, EMN:** $\xi[A, \varepsilon(\Delta^{\Box})] = \chi[A, \varepsilon(\Delta^{\Diamond 0})] = \emptyset.$

(ii) **EC, ECN, EMC, EMCN:** $\xi[A, \varepsilon(\Delta^{\Box}), \varepsilon(\Delta^{\Diamond 0})] = \emptyset.$

(iii) **EN, ECN, EMN, EMCN:** *there is an A–saturated set* Δ'.

If $\Delta \vdash \Diamond A$, *then:*

(iv) **E, EN:** $\xi[A, \varepsilon(\Delta^{\Diamond 0})] = \chi[A, \varepsilon(\Delta^{\Box})] = \emptyset.$

(v) **EM, EMN:** $\zeta[A, \varepsilon(\Delta^{\Diamond 0})] = \eta[A, \varepsilon(\Delta^{\Box})] = \emptyset.$

(vi) **EC, ECN, EMC, EMCN:** $\chi[A, \varepsilon(\Delta^{\Diamond 0}), \varepsilon(\Delta^{\Box})] = \emptyset.$

(vii) **EN, ECN, EMN, EMCN:** *there is a* $\neg A$*–saturated set* Δ'.

Proof.

(i) Suppose $\Delta \nvdash \Box A$, and let $B \in \xi[A, \varepsilon(\Delta^{\Box})]$. So $\Box B \in \Delta$. We then have:

(i^a) [**E, EN**] Since $B \approx_S A$, $B \leftrightarrow A$ is a member of every saturated set, so $B \leftrightarrow A \in \Delta$, and $\Box B \leftrightarrow \Box A \in \Delta$.

(i^b) [**EM, EMN**] Since $B \approx>_S A$, $B \rightarrow A$ is a member of every saturated set, so $B \rightarrow A \in \Delta$, $\Box B \rightarrow \Box A \in \Delta$; and then $\Box A \in \Delta$.

But then, in both cases, it cannot be that $\Delta \nvdash \Box A$. So $\xi[A, \varepsilon(\Delta^{\Box})] = \emptyset$. Now let $C \in \chi[A, \varepsilon(\Delta^{\Diamond 0})]$. Thus:

(i^c) [**E, EN**] $C \not\approx_S A$, and $\Diamond C \notin \Delta$. Hence $C \approx \neg A$. So $\vdash C \leftrightarrow \neg A$; $\vdash \Diamond C \leftrightarrow \Diamond \neg A$; $\Diamond C \leftrightarrow \Diamond \neg A \in \Delta$. Since $\Diamond C \notin \Delta$, $\Diamond \neg A \notin \Delta$, so $\neg \Diamond \neg A \in \Delta$. Since $\vdash \Box A \leftrightarrow \neg \Diamond \neg A$, $\Box A \in \Delta$,

(i^d) [**EM, EMN**] $C <\not\approx_S A$, $\Diamond C \notin \Delta$. Hence $\neg C \approx>_S A$. So $\neg C \rightarrow A$ is a member of every saturated set, so $\Box \neg C \rightarrow \Box A \in \Delta$. Since $\Diamond C \notin \Delta$, $\neg \Box \neg C \notin \Delta$, thus $\Box \neg C \in \Delta$. It follows that $\Box A \in \Delta$.

Hence, in both cases, $\Delta \vdash \Box A$, what cannot be. It follows that $\chi[A, \varepsilon(\Delta^{\Diamond 0})] = \emptyset$.

(ii) Suppose $\Delta \nvdash \Box A$, and let $\Theta = \{\theta_1,\dots,\theta_k\}$, $\Phi = \{\varphi_1,\dots,\varphi_m\}$ such that $<\Theta, \Phi> \in \xi[A, \varepsilon(\Delta^{\Box}), \varepsilon(\Delta^{\Diamond 0})]$. Then, since $\Theta \leftrightarrow \varepsilon(\Delta^{\Box})$, $\{\Box\theta_1,\dots,\Box\theta_k\} \subseteq \Delta$. Since $\Phi \subseteq \varepsilon(\Delta^{\Diamond 0})$, $\{\Diamond\varphi_1,\dots,\Diamond\varphi_m\} \subseteq_0 \Delta$. I.e., $\{\neg\Diamond\varphi_1,\dots,\neg\Diamond\varphi_m\} \subseteq \Delta$. It follows then that $\Box\theta_1 \wedge \dots \wedge \Box\theta_k \in \Delta$, and that $\Box(\theta_1 \wedge \dots \wedge \theta_k) \in \Delta$. Similarly, $\neg\Diamond\varphi_1 \wedge \dots \wedge \neg\Diamond\varphi_m \in \Delta$, and that $\neg\Diamond\neg(\neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \in \Delta$; $\Box(\neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \in \Delta$. From axiom C, then, $\Box(\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \in \Delta$. Thus we have:

(ii^a) [EC, ECN]
Since $A \approx_S <\Theta, \Phi>$, for every saturated set Δ', $A \in \Delta'$ iff $\Theta \subseteq \Delta'$ and $\Phi \subseteq_0 \Delta'$. That is, for every saturated set Δ', $A \in \Delta'$ iff $\{\theta_1,\dots,\theta_k\} \subseteq \Delta'$ and $\{\varphi_1,\dots,\varphi_m\} \subseteq_0 \Delta'$; i.e., $\{\neg\varphi_1,\dots,\neg\varphi_m\} \subseteq \Delta'$. It follows that for every saturated set Δ', $A \in \Delta'$ iff $\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m \in \Delta'$. Thus, for every saturated set Δ', $A \leftrightarrow \theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m \in \Delta'$. By RE, $\Box A \leftrightarrow \Box(\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \in \Delta'$; $\Box A \leftrightarrow \Box(\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \in \Delta$.

(ii^b) [EMC, EMCN]
Since $<\Theta, \Phi> \approx>_S A$, for every saturated set Δ', if $\Theta \subseteq \Delta'$ and $\Phi \subseteq_0 \Delta'$ then $A \in \Delta'$. That is, for every saturated set Δ', if $\{\theta_1,\dots,\theta_k\} \subseteq \Delta'$ and $\{\varphi_1,\dots,\varphi_m\} \subseteq_0 \Delta'$ then $A \in \Delta'$; i.e., if $\{\theta_1,\dots,\theta_k\} \subseteq \Delta'$ and $\{\neg\varphi_1,\dots,\neg\varphi_m\} \subseteq \Delta'$ then $A \in \Delta'$. It follows that for every saturated set Δ', if $\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m \in \Delta'$ then $A \in \Delta'$. Thus, for every saturated set Δ', $\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m \to A \in \Delta'$. By RM, $\Box(\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \to \Box A \in \Delta'$; $\Box(\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \to \Box A \in \Delta$.
In both cases, since $\Box(\theta_1 \wedge \dots \wedge \theta_k \wedge \neg\varphi_1 \wedge \dots \wedge \neg\varphi_m) \in \Delta$, $\Box A \in \Delta$, $\Delta \vdash \Box A$, what cannot be. Thus $\xi[A, \varepsilon(\Delta^{\Box}), \varepsilon(\Delta^{\Diamond 0})] = \emptyset$.
(iii) Suppose there is no A–saturated set Δ'. So, for every Δ', $A \in \Delta'$. Let B be a theorem: so, for every saturated set Δ', $\Delta' \vdash B$, $B \in \Delta'$. Thus $B \approx_S A$. Further, since $\vdash B$, we have $\vdash \Box B$ and, for every saturated set Δ', $\Delta' \vdash \Box B$, $\Box B \in \Delta'$. But then $\Box B \in \Delta$, $B \in \varepsilon(\Delta^{\Box})$, and, in this case, $\xi[A, \varepsilon(\Delta^{\Box})] \neq \emptyset$, what cannot be. So there is an A-saturated set Δ'.

The proof of cases (iv) – (vii) is similar to cases (i) – (iii). ■

Theorem T26. *For every A–saturated set Δ and every normal sequence α_n, the characteristic function f of Δ is an α_n–valuation.*

Proof. First of ali, it is easy to prove by P24 that

(†) The characteristic function f of Δ is a semi–valuation.

We now prove the theorem by induction on n. If $n = 1$, the property follows from (†) above. Let us suppose $n > 1$.

(1) If, for every $m < n$, $A_n \neq \Box A_m$, $A_n \neq \Diamond A_m$, f is trivially an α_n–valuation.

(2) For some $m < n$, $A_n = \Box A_m$.

I) $f(A_n) = 0$. Then $A_n \notin \Delta$, $\Delta \nvdash \Box A_m$.
From L19 we have now the following:

(i^a) **E, EN, EM, EMN**: $\xi[A_m, \varepsilon(\Delta^\Box)] = \chi[A_m, \varepsilon(\Delta^{\Diamond 0})] = \emptyset$.

(i^b) **EC, ECN, EMC, EMCN**: $\xi[A_m, \varepsilon(\Delta^\Box), \varepsilon(\Delta^{\Diamond 0})] = \emptyset$.

We have then that $\{\alpha_{n-1}\}^\Box_{f,1} \subseteq \Delta^\Box$ and $\{\alpha_{n-1}\}^\Diamond_{f,0} \subseteq \Delta^{\Diamond 0}$, thus:

(ii^a) **E, EN, EM, EMN**: $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^\Box_{f,1})] = \chi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^\Diamond_{f,0})] = \emptyset$.

(ii^b) **EC, ECN, EMC, EMCN**: $\xi^{n-1}[A_m, \varepsilon(\{\alpha_{n-1}\}^\Box_{f,1}), \varepsilon(\{\alpha_{n-1}\}^\Diamond_{f,0})] = \emptyset$.

Now, in **EN, EMN, ECN** and **EMCN**, from L19 there is an A_m-saturated set Δ'. By the induction hypothesis, the characteristic function f' of Δ' is an α_{n-1}–valuation. So there is an α_{n-1}–valuation f' such that $f'(A_m) = 0$. Hence f is an α_n–valuation.

II) $f(A_n) = 1$. So $\Box A_m \in \Delta$. Let us suppose there is some p, some q, $q < p \leq n$ such that $A_p = \Box A_q$ and $f(A_p) = 0$.
From L19, we have

(i^a) **E, EN, EM, EMN**: $\xi[A_q, \varepsilon(\Delta^\Box)] = \chi[A_q, \varepsilon(\Delta^{\Diamond 0})] = \emptyset$.

(i^b) **EC, ECN, EMC, EMCN**: $\xi[A_q, \varepsilon(\Delta^\Box), \varepsilon(\Delta^{\Diamond 0})] = \emptyset$.

By the induction hypothesis, f is an α_{n-1}–valuation, so it is $\Box$–α_{n-1}–normal. Thus for every p, every q, $q < p \leq n$ such that $A_p = \Box A_q$ and $f(A_p) = 0$,

(ii^a) **E, EN, EM, EMN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^\Box_{f,1})] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^\Diamond_{f,0})] = \emptyset$.

(ii^b) **EC, ECN, EMC, EMCN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^\Box_{f,1}), \varepsilon(\{\alpha_{n-1}\}^\Diamond_{f,0})] = \emptyset$.

Now, since $f(A_n) = 1$, $\varepsilon(\{\alpha_{n-1}\}^\Box_{f,1}) \cup \{A_m\} = \varepsilon(\{\alpha_n\}^\Box_{f,1})$; and since $\{\alpha_n\}^\Box_{f,1} \subseteq \Delta^\Box$,

(iii^a) **E, EN, EM, EMN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^\Box_{f,1}) \cup \{A_m\}] = \chi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^\Diamond_{f,0})] = \emptyset$.

(iii^b) **EC, ECN, EMC, EMCN**: $\xi^{n-1}[A_q, \varepsilon(\{\alpha_{n-1}\}^\Box_{f,1}) \cup \{A_m\}, \varepsilon(\{\alpha_{n-1}\}^\Diamond_{f,0})] = \emptyset$.

Now, in **EN, EMN, ECN** and **EMCN**, from L19 there is an A_q-saturated set Δ' and, by the induction hypothesis, the characteristic function f' of Δ' is an α_{n-1}–valuation. So there is an α_{n-1}–valuation f' such that $f'(A_q) = 0$. If there is some p, some q, $q < p \leq n$ such that $A_p = \Diamond A_q$ and $f(A_p) = 1$, the proof is similar. Hence, f is an α_n–valuation.

(3) For some $m < n$, $A_n = \Diamond A_m$. Proof as in (2). ■

Corollary. *v is a valuation iff v is the characteristic function of some saturated set* Δ.

Proof. As in normal modal logics ■

Theorem T27. (*Completeness Theorem*) *If* $\Gamma \models A$ *then* $\Gamma \vdash A$.

Proof. As in normal modal logics. ■

5

GTTs for K

Brady's First Law of Problem Solving:

When confronted by a difficult problem,
you can solve it more easily by reducing it to the question,
"How would the Lone Ranger have handled this?".

Now that we have seen how a valuations semantics looks like, let us look for a way of obtaining GTTs out of it. I'm going here to take **K** as an example; changes for other logics are more or less straightforward.

If we have a wff A, it is easy to construct a finite normal sequence $A_1,...,A_n$ where A is the last term—one just takes A and its proper subformulas. Now let $V(A_1,...,A_n)$ be the class of all $A_1,...,A_n$-valuations. Let us define an equivalence relation over this class as follows: $v \cong v'$ iff, for $1 \leq i \leq n$, $v(A_i) = v'(A_i)$. Since $A_1,...,A_n$ is a finite sequence, $V(A_1,...,A_n)/_{\cong}$ is obviously finite too. Thus a decision procedure for **K** consists in a procedure which allow us to reduce, for every $|v| \in V(A_1,...,A_n)/_{\cong}$, the restriction v^* of some $v' \in |v|$ to the set $\{A_1,...,A_n\}$. Such construction, which we will designate by $T[A_1,...,A_n]$, and call the GTT for $A_1,...,A_n$, will decide on the validity of any formula belonging to the sequence (and, consequently, of the property of being a theorem). In particular, of the formula A.

Let us first examine, by means of an example, how things are supposed to work. Let A be the formula $\Box\neg\neg p\rightarrow\Box p$, where p is a propositional variable. We'll construct a normal sequence $A_1,...,A_n$ where $A_n = A$ by listing ali subformulas of A. As a result we get the sequence p, $\neg p$, $\neg\neg p$, $\Box p$, $\Box\neg\neg p$, $\Box\neg\neg p\rightarrow\Box p$, which has six elements. The procedure I am going to show consists in constructing the table for A_1—i.e., $T[A_1]$ and then extending it successively to the rest of the sequence: $T[A_1,A_2],...,$ $T[A_1,...,A_6]$. At the end, the table looks like the following:

	1	2	3	4	5	6
	p	$\neg p$	$\neg\neg p$	$\Box p$	$\Box\neg\neg p$	$\Box\neg\neg p\rightarrow\Box p$
1)	1	0	1	1	1	1
2)	0	1	0	1	0	0
3)	1	0	1	0	1	1
4)	0	1	0	0	0	0

fig. 19

$T[A_1]$, $T[A_1,A_2]$ and $T[A_1,A_2,A_3]$ are constructed in the usual way, i.e., like in classical propositional logic: you assign values to the variable(s), and then proceed by calculating the value of more complex formulas. In the picture, this correspond to lines 1) and 2), rows 1 to 3. But in $T[A_1,...,A_4]$ two new lines were added: 3) and 4). I'll try to explain why. $T[A_1,A_2,A_3]/_{\cong}$ has obviously just two elements: $|v_1|$ and $|v_2|$. The elements of $|v_1|$ and $|v_2|$, restricted to A_1,A_2,A_3 are represented in the table by lines 1) and 2). Now, since $\{A_1,A_2,A_3\}^{\Box} = \{A_1,A_2,A_3\}^{\Diamond} = \emptyset$, we have, for $i \in \{1,2\}$:

I) $v_1(p) = 0$, $v_1 \vDash_1 \varepsilon(\{A_1,A_2,A_3\}^{\Box}{}_{v_i,1})$, $v_1 \vDash_0 \varepsilon(\{A_1,A_2,A_3\}^{\Diamond}{}_{v_i,0})$, i.e., $v_i{<}3{>}v_1$. In this way are fulfilled the necessary conditions which justify the existence of v' and v'' in $T[A_1,..,A_4]$ such that $v' \in |v_1|$, $v'' \in |v_2|$ and $v'(A_4) = v''(A_4) = 0$.

II) vacuously, for every p, every q, $q < p \leq 4$ such that $A_p = \Box A_q$ and $v_i(A_p) = 0$ [$A_p = \Diamond A_q$ and $v_i(A_p) = 1$], there is a $j \in \{1,2\}$ such that $v_j(A_q) = 0$ [$= 1$], $v_j(A_4) = 1$ and $v_i{<}3{>}v_j$. So are fulfilled the necessary conditions for the existence of v^*, v^{**} in $T[A_1,...,A_4]$ such that $v^* \in |v_1|$, $v^{**} \in |v_2|$ and $v^*(A_4) = v^{**}(A_4) = 1$.

By I) and II), it is plausible that $T[A_1,...,A_4]/_{\cong}$ has thus four elements $|v_1|...|v_4|$, such that the restriction of each one to $A_1,..,A_4$ is represented by lines 1) to 4) of the GTT. This is how and why we got these two extra lines.

Such an unfolding, now, doesn't happen in the construction of $T[A_1,...,A_5]$—which was to be expected, since we p and $\neg\neg p$ are equivalent, and in consequence $\Box p$ and $\Box\neg\neg p$ should get the same value. Let us see, for instance, how $\Box\neg\neg p$ cannot possibly take the value 0 in line 1). Let $v \in |v_1|$, whose restriction to $A_1,...,A_4$ is represented by line 1). We have that $\{A_1,...,A_4\}^{\Diamond}{}_{v,0} = \emptyset$ and $\{A_1,...,A_4\}^{\Box}{}_{v,1} = \{\Box p\}$; i.e., $\varepsilon(\{A_1,...,A_4\}^{\Box}{}_{v,1}) = \{p\}$. If we had $v(\Box\neg\neg p) = 0$, we should have a v_i, $i \in \{1,...,4\}$ such that $v_i(\neg\neg p) = 0$ and $v{<}4{>}v_i$—i.e., $v_i \vDash_1 \varepsilon(\{A_1,...A_4\}^{\Box}{}_{v,1})$, so $v_i(p) = 1$. But $v_i(\neg\neg p) = 0$ and $v_i(p) = 1$ is in every valuation (and so in every line) an impossibility; hence $\Box\neg\neg p$ cannot get a 0 in line 1).

On the other hand we can see that $v(\Box\neg\neg p) = 1$ is possible, for, vacuously, for every p, every q, $q < p < 5$, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is a $j \in \{1,...,4\}$ such that $v_j(A_q) = 0$, $v_j(A_3) = 1$ and $v{<}4{>}v_j$.

This situation also happens in line 2), where once again $\Box\neg\neg p$ cannot take the value 0, only 1. In lines 3) and 4) it's the other way round: $\Box\neg\neg p$ can only take a 0. So these are the reasons why we don't get a splitting of lines in the construction of $T[A_1,...,A_5]$. And by $T[A_1,...,A_6]$ we are again in the realm of things classical.

I hope that this example has helped to make things a little bit more clear, because now we will have to define everything rigorously, and this is far from being easy—the definition of a GTT is a very big one. I'll state the definition first and give some explanations later on.

Definition D15. Let α_n be a normal sequence. A *generalized truth-table* (GTT) for α_n is a function $T[\alpha_n] : \{\alpha_n\} \times J(\alpha_n) \rightarrow \{0,1\}$, where:

1) for $n = 1$, $J(\alpha_1) = \{1,2\}$, $T[\alpha_1](A_1, 1) = 1$ and $T[\alpha_1](A_1, 2) = 0$;

2) for $n > 1$, and $J(\alpha_{n-1}) = \{1,...,q\}$:

(a) if A_n is a propositional variable, then $J(\alpha_n) = \{1,...,2q\}$ and:

i) for $i < n$, $j \in J(\alpha_{n-1})$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i < n$, $j' \in J(\alpha_{n-1})$ and $j = q + j'$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j')$;

iii) for $i = n, j \in \mathrm{J}(\alpha_{n-1})$, $T[\alpha_n](A_i, j) = 1$;

iv) for $i = n, j' \in \mathrm{J}(\alpha_{n-1})$ and $j = q + j'$, $T[\alpha_n](A_i, j) = 0$;

(b) if $A_n = \neg A_k$, $k < n$, $\mathrm{J}(\alpha_n) = \mathrm{J}(\alpha_{n-1})$ and:

i) for $i < n$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i = n$, $T[\alpha_n](A_i, j) \neq T[\alpha_{n-1}](A_k, j)$;

(c) if $A_n = A_k \rightarrow A_e$, $k < n$, $e < n$, $\mathrm{J}(\alpha_n) = \mathrm{J}(\alpha_{n-1})$ and:

i) for $i < n$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i = n$, $T[\alpha_n](A_i, j) = 1$ iff $T[\alpha_{n-1}](A_k, j) = 0$ or $T[\alpha_{n-1}](A_e, j) = 1$;

(d) if $A_n = \Box A_k$, $k < n$, then for every $j \in \mathrm{J}(\alpha_{n-1})$:

I) let $\alpha(j, n\text{-}1) = \{j' \in \mathrm{J}(\alpha_{n-1})$: $T[\alpha_{n-1}](A_k, j') = 0$ and, for every r, $1 \leq r \leq n$, if $A_r = \Box A_s$ and $T[\alpha_{n-1}](A_r, j) = 1$ $[A_r = \Diamond A_s$ and $T[\alpha_{n-1}](A_r, j) = 0]$, then $T[\alpha_{n-1}](A_s, j') = 1$ $[= 0]\}$;

II) for every p, every q, $q < p < n$ such that $A_p = \Box A_q$ and $T[\alpha_{n-1}](A_p, j) = 0$ $[A_p = \Diamond A_q$ and $T[\alpha_{n-1}](A_p, j) = 1]$, let $\beta(p, j, n\text{-}1) = \{j' \in \mathrm{J}(\alpha_{n-1})$: $T[\alpha_{n-1}](A_q, j') = 0$ $[= 1]$, $T[\alpha_{n-1}](A_k, j') = 1$ and, for every r, $1 \leq r \leq n$, if $A_r = \Box A_s$ and $T[\alpha_{n-1}](A_r, j) = 1$ $[A_r = \Diamond A_s$ and $T[\alpha_{n-1}](A_r, j) = 0]$, then $T[\alpha_{n-1}](A_s, j') = 1$ $[= 0]\}$;

III) let $\{j_1, \ldots, j_m\} \subseteq \mathrm{J}(\alpha_{n-1})$ such that:

1) $j_{m'} < j_{m''}$ if $m' < m''$;

2) $j_{m'} \in \{j_1, \ldots, j_m\}$ if $\alpha(j_{m'}, n\text{-}1) \neq \emptyset$ and, for every $p < n$, $\beta(p, j_{m'}, n\text{-}1) \neq \emptyset$.

Then $\mathrm{J}(\alpha_n) = \{1, \ldots, q, \ldots, q+m\}$ and:

i) for $i < n, j \leq q$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i < n, j = q + m'$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j_{m'})$;

iii) for $i = n$, j such that $\alpha(j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 1$;

iv) for $i = n$, j such that $\alpha(j, n\text{-}1) \neq \emptyset$ and, for some $p < n$, $\beta(p, j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 0$;

v) for $i = n$, j such that $\alpha(j, n\text{-}1) \neq \emptyset$ and, for every $p < n$, $\beta(p, j, n\text{-}1) \neq \emptyset$, in which case, for some $m' \in \{1, \ldots, m\}$, $j = j_{m'}$ or $j = q + m'$,

1) if $j = j_{m'}$ then $T[\alpha_n](A_i, j) = 1$;

2) if $j = q + m'$ then $T[\alpha_n](A_i, j) = 0$.

(e) if $A_n = \Diamond A_k$, $k < n$, then for every $j \in \mathrm{J}(\alpha_{n-1})$:

I) let $\gamma(j, n\text{-}1) = \{j' \in \mathrm{J}(\alpha_{n-1})$: $T[\alpha_{n-1}](A_k, j') = 1$ and, for every r, $1 \leq r \leq n$, if $A_r = \Diamond A_s$ and $T[\alpha_{n-1}](A_r, j) = 0$ $[A_r = \Box A_s$ and $T[\alpha_{n-1}](A_r, j) = 1]$, then $T[\alpha_{n-1}](A_s, j') = 0$ $[= 1]\}$;

II) for every p, every q, $q < p < n$ such that $A_p = \Diamond A_q$ and $T[\alpha_{n-1}](A_p, j) = 1$ $[A_p = \Box A_q$ and $T[\alpha_{n-1}](A_p, j) = 0]$, let $\delta(p, j, n\text{-}1) = \{j' \in \mathrm{J}(\alpha_{n-1})$: $T[\alpha_{n-1}](A_q, j') = 1$ $[= 0]$, $T[\alpha_{n-1}](A_k, j') = 0$ and, for every r, $1 \leq r \leq n$, if $A_r = \Diamond A_s$ and $T[\alpha_{n-1}](A_r, j) = 0$ $[A_r = \Box A_s$ and $T[\alpha_{n-1}](A_r, j) = 1]$, then $T[\alpha_{n-1}](A_s, j') = 0$ $[= 1]\}$;

III) let $\{j_1, \ldots, j_m\} \subset \mathrm{J}(\alpha_{n-1})$ such that:

1) $j_{m'} < j_{m''}$ if $m' < m''$;

2) $j_{m'} \in \{j_1, \ldots, j_m\}$ if $\gamma(j_{m'}, n\text{-}1) \neq \emptyset$ and, for every $p < n$, $\delta(p, j_{m'}, n\text{-}1) \neq \emptyset$.

Then $\mathrm{J}(\alpha_n) = \{1, \ldots, q, \ldots, q+m\}$ and:

i) for $i < n, j \leq q$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i < n, j = q + m'$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j_{m'})$;

iii) for $i = n$, j such that $\gamma(j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 0$;

iv) for $i = n$, j such that $\gamma(j, n\text{-}1) \neq \emptyset$ and, for some $p < n$, $\delta(p, j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 1$;

v) for $i = n, j$ such that $\gamma(j, n\text{-}1) \neq \emptyset$ and, for every $p < n$, $\delta(p, j, n\text{-}1) \neq \emptyset$, in which case, for some $m' \in \{1,...,m\}, j = j_{m'}$ or $j = q + m'$,
1) if $j = j_{m'}$ then $T[\alpha_n](A_i, j) = 0$;
2) if $j = q + m'$ then $T[\alpha_n](A_i, j) = 1$.

Now to the explanations of this all. Intuitively, a GTT is a function with two arguments, the first being a formula (in a normal sequence), what would correspond to a row in a "normal" table, and the second pointing to a line: thus for instance we can express the fact that in our example the formula $\Box p$ gets the value 0 on line 3) by stating that $T[A_1,...,A_4](\Box p, 3) = 0$. Of course, as the table expands, the number of rows and lines increases, but formulas in the expanded table preserve the value they already have. Thus $T[A_1,...,A_6](\Box p, 3) = T[A_1,...,A_5](\Box p, 3) = 0$.

Back to the explanations. $J(\alpha_n)$ denotes the set of lines of the GTT. When $n = 1$ we have a normal sequence with just one element, which must in this case be a propositional variable (or the sequence wouldn't be normal). So $J(\alpha_1) = \{1,2\}$: that is, we have two lines in our table, and the variable gets value 1 in line 1) and 0 in line 2).

Clause a) just states the fact that when we found another propositional variable, we must double the number of lines (from $\{1,...,q\}$ to $\{1,...,2q\}$). The new variable gets value 1 in lines 1 up to q, and 0 from $q+1$ up to $2q$.

Clauses b) and c) offer no problem: the new function $T[A_1,...,A_n]$ get the same values as $T[A_1,...A_{n\text{-}1}]$ for wffs whose index is smaller than n; for A_n the classical conditions must be preserved: so for instance in b), for $i = n, A_n = \neg A_k$ gets the opposite value of A_k. Similarly for implication.

Let us then consider clause d), where we handle the case where $A_n = \Box A_k$, for some $k < n$. First we define for each line j a certain sets of lines called $\alpha(j, n\text{-}1)$—this is just the set of those lines j' who give 0 to A_k and which satisfy all A_i such that $\Box A_i$ have value 1 in line j, and reject all A_i such that $\Diamond A_i$ have value 0 in line j. In a similar way we define, for each p such that for some q, $A_p = \Box A_q$ [$= \Diamond A_q$] and which gets value 0 [1] on line j, the set $\beta(p, j, n\text{-}1)$.

Now to III): $\{j_1,...j_m\}$ marks a subset of the set $J(\alpha_{n\text{-}1})$ of lines, namely those lines which potentially lead to splittings: that is, there are two possible ways of extending them, one in which $\Box A_k$ gets 1, the other in which it gets 0. These splitting character of some lines comes from the fact that their α-sets are not empty (i.e., some line j' gives 0 to A_i and satisfies/rejects the scope of j), the same holding of their β-sets: for every p which is a false necessity there is a line ensuring that and giving 1 to A_i. Now m is the number of lines which split, so the table gets m extra lines: $J(\alpha_n) = \{1,...,q,...,q+m\}$.

The formulas whose index in the sequence is less than i conserve their values in the extension: clauses i) and ii) (for $i < n$). Clause iii) states that, if no line satisfying the scope of line j gives 0 to A_k, then $\Box A_k$ gets 1 in j. In clause iv) there is such a line, so $\Box A_k$ gets 0, and, since the β-set is empty, for some p, 0 is the only possibility. Compare this to clause v): there both sets are non-empty, so the line has to split. The "old" line ($j_{m'}$) gives 1 to $\Box A_k$, and the new one (j_{q+m}), the value 0.

All clear? Then let us try to prove some results about this all.

Lemma L20. *Let α_n be a normal sequence. For every $j \in J(\alpha_k)$, $1 \leq k \leq n$, there is $j^* \in J(\alpha_n)$ such that, for every i, $1 \leq i \leq k$, $T[\alpha_k](A_i, j) = T[\alpha_n](A_i, j^*)$.*

Proof. This lemma can be easily proved by induction on $n-k$, based on conditions i) and ii) of D15.1.a through D15.1.d.

Lemma L21. *Let α_n be a normal sequence, and v a valuation. Then there is $j \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j)$.*

Proof. By induction on n. Let v be a valuation.

(1) $n = 1$. By D15.1, $J(\alpha_1) = \{1, 2\}$. There are two possibilities:

I) $v(A_1) = 0$. From D15.1.1 we have that $T[\alpha_1](A_1, 2) = 0$. Hence there is $j = 2$ such that $v(A_1) = T[\alpha_1](A_1, j)$.

II) $v(A_1) = 1$. From D15.1.1 we have that $T[\alpha_1](A_1, 1) = 0$. Hence there is $j = 1$ such that $v(A_1) = T[\alpha_1](A_1, j)$.

(2) Let $n > 1$, and let $J(\alpha_{n-1}) = \{1,\dots,q\}$.

Induction hypothesis: for every valuation v there is $j \in J(\alpha_{n-1})$ such that, for $1 \leq i < n$, $v(A_i) = T[\alpha_{n-1}](A_i, j)$. That is, for every valuation v there is a line j of the table until n-1, such that j and v agree for $i < n$. We need now to prove there is a line j such that v and j agree until n, what we'll do examining the construction of the GTT.
By D15.1, for $i < n$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$. It follows from L21 that:

(†) For every valuation v there is $j \in J(\alpha_n)$ such that, for $1 \leq i < n$, $v(A_i) = T[\alpha_n](A_i, j)$.

(a) Let A_n be a propositional variable. By D15.2.a, $J(\alpha_n) = \{1,\dots,2q\}$.

I) $v(A_n) = 0$. From (†) there is $j \in J(\alpha_n)$ such that, for $1 \leq i < n$, $v(A_i) = T[\alpha_n](A_i, j)$. Let us suppose that $j \in J(\alpha_{n-1})$—i.e., $j \in \{1,\dots,q\}$. By D15.2.a.ii, there is $j^* \in J(\alpha_n)$ such that, for $i < n$, $T[\alpha_n](A_i, j^*) = T[\alpha_n](A_i, j)$; hence, for $i < n$, $v(A_i) = T[\alpha_n](A_i, j^*)$. By D15.2.a.iv, for $i = n$, $T[\alpha_n](A_i, j^*) = 0$. Hence there is $j^* \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $T[\alpha_n](A_i, j^*) = v(A_i)$. Let us now suppose that $j = q + j'$. By D15.2.a.iv, $T[\alpha_n](A_n, j) = 0$—thus there is $j \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j)$.

II) $v(A_n) = 1$. Proof as in case I).

(b) Let $A_n = \neg A_k$, for some $k < n$. By D15.2.b, $J(\alpha_n) = J(\alpha_{n-1})$.
I) $v(A_n) = 0$. Then $v(A_k) = 1$, since v is also a semi-valuation. By (†) there is $j \in J(\alpha_n)$ such that $T[\alpha_n](A_k, j) = v(A_k) = 1$, By D15.2.b.i, $T[\alpha_n](A_k, j) = T[\alpha_{n-1}](A_k, j)$, and, by D15.2.b.ii, $T[\alpha_n](A_n, j) \neq T[\alpha_{n-1}](A_k, j)$. That is, $T[\alpha_n](A_n, j) = 0$. Hence there is $j \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j)$.

II) $v(A_n) = 1$. Proof as in case I).

(c) Let $A_n = A_k \to A_e$, for some $e, k < n$. By D15.2.b, $J(\alpha_n) = J(\alpha_{n-1})$.
I) $v(A_n) = 0$. Then $v(A_k) = 1$ and $v(A_e) = 0$, since v is also a semi-valuation. By (†) there is $j \in J(\alpha_n)$ such that $T[\alpha_n](A_k, j) = 1$ and $T[\alpha_n](A_e, j) = 0$. By D15.2.b.i, $T[\alpha_n](A_k, j) = T[\alpha_{n-1}](A_k, j)$, $T[\alpha_n](A_e, j) = T[\alpha_{n-1}](A_e, j)$. By D15.2.b.ii, $T[\alpha_n](A_n, j) = 0$. Hence there is $j \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j)$.

II) $v(A_n) = 1$. Proof as in case I).

(d) $A_n = \Box A_m$, for some $m < n$.

(I) $v(A_n) = 0$. By definition, v is an α_n–valuation, and thus there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v<n\text{-}1>v_n$. By (†) there is $j_n \in J(\alpha_n)$ such that, for $i < n$, $T[\alpha_n](A_i, j_n) = v_n(A_i)$. Then $T[\alpha_n](A_m, j_n) = 0$ and, for every r, $1 \leq r < n$ such that $A_r = \Box A_s$ and $v(A_r) = 1$ [$A_r = \Diamond A_s$ and $v(A_r) = 0$], $T[\alpha_n](A_r, j) = 1$ [$= 0$] and $T[\alpha_n](A_s, j_n) = 1$ [$= 0$]. By definition, $\alpha(j, n\text{-}1) \neq \emptyset$. If we now check D15.2.d.iv and D15.2.d.v we see that, whatever the case, there is always a j^* such that, for $i < n$, $T[\alpha_n](A_i, j) = T[\alpha_n](A_i, j^*)$, and, besides, $T[\alpha_n](A_n, j^*) = 0$. Hence there is $j^* \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j^*)$.

(II) $v(A_n) = 1$. By definition, v is an α_n–valuation, and thus for every p, every q, $q < p \leq n$ such that $A_p = \Box A_q$ and $v(A_p) = 0$ [$A_p = \Diamond A_q$ and $v(A_p) = 1$], there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$ [$= 1$], $v_p(A_m) = 1$ and $v<n\text{-}1>v_p$. By (†) there is $j \in J(\alpha_n)$ such that, for $i < n$, $T[\alpha_n](A_i, j) = v(A_i)$. That is, $T[\alpha_n](A_p, j) = 0$ [$= 1$]. Besides, for every r, $1 \leq r < n$ such that $A_r = \Box A_s$ and $v(A_r) = 1$ [$A_r = \Diamond A_s$ and $v(A_r) = 0$], $T[\alpha_n](A_r, j) = 1$ [$= 0$]. Also by (†) there is $j_n \in J(\alpha_n)$ such that, for $i < n$, $T[\alpha_n](A_i, j_n) = v_n(A_i)$. That is, $T[\alpha_n](A_q, j_n) = 0$ [$= 1$], $T[\alpha_n](A_m, j_n) = 1$ and, for every r, $1 \leq r < n$ such that $A_r = \Box A_s$ and $v(A_r) = 1$ [$A_r = \Diamond A_s$ and $v(A_r) = 0$], $T[\alpha_n](A_s, j_n) = 1$ [$= 0$]. Thus it follows that, for every $p < n$, that $\beta(p, j, n\text{-}1) \neq \emptyset$. By D15.2.d.iv, there $j \in J(\alpha_n)$ ($j = j_{m'}$) such that, for $1 \leq i < n$, $v(A_i) = T[\alpha_n](A_i, j)$ and $T[\alpha_n](A_n, j) = 1$. Thus, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j)$.

(e) $A_n = \Diamond A_m$, for some $m < n$. Proof as in (d). ■

Lemma L22. *For every normal sequence* α_n, *if, for some* $i \leq n$, $\vdash_K$ A$_i$ *then for every* $j \in J(\alpha_n)$, $T[\alpha_n](A, j) = 1$.

Proof. By induction on the number r of lines of a proof of A_i in K.

(1) $r = 1$. In this case, A_i is an axiom.

(a) Let $A_i = A \rightarrow (B \rightarrow A)$. Let us suppose that, for some $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 0$. By D15.2.c.ii, $T[\alpha_n](A, j) = 1$ and $T[\alpha_n](B \rightarrow A, j) = 0$. Again by D15.2.c.ii, $T[\alpha_n](B, j) = 1$ and $T[\alpha_n](A, j) = 0$, what is not possible. Thus for every $j \in J(\alpha_n)$, $T[\alpha_n](A, j) = 1$.

(b) If now A_i is an instance of $A2$ or $A3$, the proof is analogous as in (a).

(c) Let $A_i = \Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$. Let us suppose that, for some $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 0$. By D15.2.c.ii, $T[\alpha_n](\Box(A \rightarrow B), j) = T[\alpha_n](\Box A, j) = 1$ and $T[\alpha_n](\Box B, j) = 0$. By D15.2.d we see that, since $T[\alpha_n](\Box B, j) = 0$, that $\alpha(j, n\text{-}1) \neq \emptyset$ (if not, by D15.2.d.iii, we would have $T[\alpha_n](\Box B, j) = 1$). From this fact it follows that there is $j_n \in J(\alpha_n)$ such that $T[\alpha_n](B, j_n) = 0$ and, for every r, $1 \leq r < n$ such that $A_r = \Box A_s$ and $T[\alpha_n](A_r, j) = 1$ [$A_r = \Diamond A_s$ and $T[\alpha_n](A_r, j) = 0$], $T[\alpha_n](A_s, j_n) = 1$ [$= 0$]. Well, $T[\alpha_n](\Box(A \rightarrow B), j) = T[\alpha_n](\Box A, j) = 1$, thus $T[\alpha_n](A \rightarrow B, j_n) = T[\alpha_n](A, j_n) = 1$, and it cannot be that $T[\alpha_n](B, j_n) = 0$. Thus for every $j \in J(\alpha_n)$, $T[\alpha_n](A, j) = 1$.

(2) $r > 1$. In this case, either A_i is an axiom, and the property is already proved in 1), or:

(a) A_i was obtained by MP from B and $B \rightarrow A_i$. We thus have that $\vdash B$ and $\vdash B \rightarrow A_i$. Let us take a normal sequence α_{n+k}, $k \geq 0$, where $A_{n+k} = B \rightarrow A_i$. Of course B occurs in this sequence. Since $\vdash B$ and $\vdash B \rightarrow A_i$, we have by the induction hypothesis that, for every $j^* \in J(\alpha_{n+k})$, $T[\alpha_{n+k}](B \rightarrow A_i, j^*) =$

$T[\alpha_{n+k}](B, j^*) = 1$. It follows that, for every $j^* \in J(\alpha_{n+k})$, $T[\alpha_{n+k}](A_i, j^*) = 1$. Now, by L20, if, for some $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 0$, then, for some $j^* \in J(\alpha_{n+k})$, $T[\alpha_{n+k}](A_i, j^*) = 0$, and this cannot be. Thus for every $j \in J(\alpha_n)$, $T[\alpha_n](A, j) = 1$.

(b) $A_i = \Box A_k$ and was obtained by *RN* from A_k. We thus have that $\vdash A_k$ and, by the induction hypothesis, for every $j \in J(\alpha_n)$, $T[\alpha_n](A_k, j) = 1$. But then, by D15.2.d.I, for every $j \in J(\alpha_n)$, $\alpha(j, n\text{-}1) = \emptyset$. Thus, by D15.2.d.iii, for every $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 1$. ■

Theorem T28. $\vdash A$ *iff for every normal sequence* α_n, *where* $A = A_i$, $1 \leq i \leq n$, *and for every* $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 1$.

Proof. A) If $\vdash A$ the proof follows immediately from L22.
B) Let us suppose that for every normal sequence α_n, where $A = A_i$, $1 \leq i \leq n$, and for every $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 1$. Suppose that $\nvdash A_i$. Then $\nvDash A_i$; thus, there is some valuation v such that $v(A_i) = 0$. By L21 there is a $j \in J(\alpha_n)$ such that $T[\alpha_n](A_i, j) = 0$, against the hypothesis. Thus $\vDash A_i$. ■

As we see, **K** is decidable by GTTs. And, as the reader certainly noticed, the definition of the GTT was exactly "copied" from the definition of an α_n–valuation. Defining GTTs for the other logics, thus, is not that difficult. I am not going to do it here for reasons of space and because it is really straightforward. (It is a good exercise, though.) [44]

[44] We are, however, going to take a look at GTTs for S4 and for the EDL Z5, from which definitions one can have an idea of, for instance, how reflexivity or euclideanity are handled.

6 The S4 problem

Nobody knows the troubles I've seen...

6.1 The problem...

In this chapter we are going to consider, albeit briefly, the problems that valuation semantics have with some normal systems of modal logic. Actually S4—i.e., **KT4**— is not alone the problem, as the title of this chapter could suggest, but, since it is the most known logic among the problematic cases, it takes the blame. To tell the truth, the problem concerns axiom schema *4*; so many systems containing this axiom are bound to make trouble. (This also includes, by the way, intuitionistic logic, which is not surprising at ali when one thinks that there is a translation function relating it and **KT4**.)

As I said in a previous chapter, there is a "natural" definition of an α_n-valuation for **KT4**. I will show how it looks like, but, for simplicity reasons, I'll let possibilites out of the picture. It goes like this, if we consider the case where $A_n = \Box A_m$:

I) if $v(A_n) = 0$, there is an α_{n-1}-valuation v_n such that $v_n(A_m) = 0$ and $v_n \models_1 \{\alpha_{n-1}\}^{\Box}{}_{v,1}$;

II) if $v(A_n) = 1$, then $v(A_m) = 1$ and for every p, every q, such that $A_p = \Box A_q$ and $v(A_p) = 0$, there is an α_{n-1}-valuation v_p such that $v_p(A_q) = 0$, $v_p(A_m) = 1$ and $v_p \models_1 \{\alpha_{n-1}\}^{\Box}{}_{v,1}$.

Now why is this definition natural? Well, for the one part, it takes care of the characteristic axiom of **KT4**; that is, axiom schema *4*. Suppose $v(\Box A \rightarrow \Box\Box A) = 0$. Then $v(\Box A) = 1$ and $v(\Box\Box A) = 0$. By the definition, there is an α_{n-1}-valuation v_n such that $v_n(\Box A) = 0$ and $v_n \models_1 \{\alpha_{n-1}\}^{\Box}{}_{v,1}$. But, since $\Box A \in \{A_1,\ldots,A_{n-1}\}^{\Box}{}_{v,1}$, we also should have $v_n(\Box A) = 1$, and this is a contradiction. So $v(\Box A \rightarrow \Box\Box A) = 1$. It is easy to see that this definition renders also the other modal axioms true.

So it seems that this definition would take care of **KT4**. But it is not the case, as far as I can tell. The problem is, one cannot prove—to be hones, I couldn't prove, with this definition, the normality lemma, that is, that α_n-valuations are $\Box_0$–α_n-normal. The proof comes to a halt because in **KT4** we have, as a derived inference rule, the following one:

$$\frac{\Box A \rightarrow B}{\Box A \rightarrow \Box B.}$$

Suppose now we have the normal sequence $A, B, \Box B, \Box A$; and let us suppose that, for every α_n-valuation v, if $v(\Box A) = 1$ then $v(B) = 1$. Let us suppose further that there is some v, such that $v(\Box A) = 1$ and $v(\Box B) = 0$. In the example, our $A_p = \Box B$, and $A_n = \Box A$. By clause II) of the definition, we should have an $\alpha_{n\text{-}1}$-valuation v_p such that $v_p(B) = 0$, $v_p(A) = 1$ and $v_p \vDash_1 (\alpha_{n\text{-}1})^{\Box}_{v,1}$. But this is not sufficient to derive a contradiction! In fact, we would need the following:

(•) there is an α_n-valuation v_p such that $v_p(B) = 0$, $v_p(A) = 1$ and $v_p \vDash_1 (\alpha_n)^{\Box}_{v,1}$.

If that were the case, we would have v_p as an α_n-valuation, and we would have $v_p(\Box A) = 1$. Since the hypothesis was that for every α_n-valuation v, if $v(\Box A) = 1$ then $v(B) = 1$, then we would have $v_p(B) = 1$, a contradiction. The rule would be validity-preserving.

However, writing the definition this way is obviously something we are not allowed to do, since it would amount to trying to define α_n-valuations by means of themselves.

A similar problem occurs with axiom schema *5*, and also because we then have, as a derived inference rule, the following one:

$$\frac{\Diamond A \to B}{\Diamond A \to \Box B.}$$

The reasons why the natural definition (for *5* alone, like in **K5**) doesn't work are pretty much the same.

Why, then, do we have valuations semantics for logics such as **KT5**, **K45**, **KD45**? Well, in these logics the requirements are that, for instance, if $f{<}k{>}g$, then $(\alpha_k)^{\Box}_{f,1} = (\alpha_k)^{\Box}_{g,1}$; and $(\alpha_k)^{\Diamond}_{f,0} = (\alpha_k)^{\Diamond}_{g,0}$. This is enough to guarantee that things work. (See proof of the Normality Lemma L11, particularly in the case of the mentioned systems.)

But maybe we can find a way out of the KT4 predicament. Let us think a bit about KT4-models, and let us consider some world in it: call it 0. Because the accessibility relation is transitive, we notice that every world x which occurs "under" 0—that is, which is accessible to 0—has either the same number of true necessities as 0, or more. It doesn't happen that in x has less necessities are true than in 0. Witness the following example picture (black circles denote true necessities):

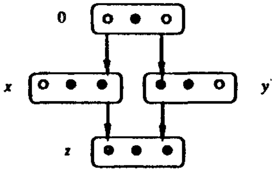

fig. 20

As one can see, the set of true necessities increases. One can also see that there is a world, namely z, which gives truth to every necessary wff (at least, to every one of the three here represented). That is, if we consider only a *finite* set of formulas, there is a point where every new accessible world ***has the same set of true necessities as the one to which it is accessible.*** This would be a world with a "maximal" number of true necessities.

6.2 ... and a solution

It is now clear how this characteristic could help us: with valuation semantics, or GTTs, for that matter, we are also working with a finite set of formulas. So we would also have some "maximal" α_n–valuation, or a "maximal" line in a table. Now to solve the **KT4** problem we could make an induction within our already inductive definition: in the case of modalized formulas, we begin by defining the maximal ones—the basic case—and then proceed "upwards" to the other ones, until we reach the levei zero.

In this chapter, we will try to put this idea into practice. Since our main interest is the construction of GTTs, I'll skip the definitions of valuation semantics and go directly to the GTT definition itself, after what we'll prove that a wff is a theorem of **KT4** iff it gets 1 in every line of the table.

A next remark is: for reasons of simplicity, we'll let the possibilities out of the picture. But the method here described can be extended to cover them too.

I will introduce two conventions. Let α_n be a normal sequence. For $1 < i \leq n$, we say that τ_i denotes the cardinality of $\{\alpha_{i-1}\}^{\Box}$, if $A_i = \Box Aj$, for $j < i$. If $i = 1$, or if $A_i \neq \Box Aj$, then $\tau_i = 0$. The second convention—actually an abbreviation—has to do with the GTTs we are going to define below. Let j be some line of of a GTT for **KT4**: we will use $\tau_i(j)$ to denote the cardinality of the subset of $\{\alpha_{i-1}\}^{\Box}$ consisting of those wffs which are given the value 1 in line j. In other words, $\tau_i(j)$ denotes the number of necessary wffs, for $k < i$, which are true in line j.

Now we can go to our definition. But, instead of a GTT being a function $T[\alpha_n]$, it will now be $T[\alpha_n, l]$, where the l parameter takes values from τ_n (maximum) down to zero.

Definition D16. Let α_n be a normal sequence. A *generalized truth-table* (GTT) for α_n is a function $T[\alpha_n, l] : \{\alpha_n\} \times J^l(\alpha_n) \rightarrow \{1, {*}, 0\}$, where:

1) for $n = 1$, $l = 0$, $J^l(\alpha_1) = \{1,2\}$, $T[\alpha_1, l](A_1, 1) = 1$ and $T[\alpha_1, l](A_1, 2) = 0$;

2) for $n > 1$, and $J^0(\alpha_{n-1}) = \{1,\ldots,q\}$:

(a) if A_n is a propositional variable, then $\tau_n = l = 0$, $J^l(\alpha_n) = \{1,\ldots,2q\}$, and:

i) for $i < n, j \in J^0(\alpha_{n-1})$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j)$;

ii) for $i < n, j' \in J^0(\alpha_{n-1})$ and $j = q + j'$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j')$;

iii) for $i = n, j \in J^0(\alpha_{n-1})$, $T[\alpha_n, l](A_i, j) = 1$;

iv) for $i = n, j' \in J^0(\alpha_{n-1})$ and $j = q + j'$, $T[\alpha_n, l](A_i, j) = 0$;

(b) if $A_n = \neg A_k$, $k < n$, $\tau_n = l = 0$, $J^l(\alpha_n) = J^0(\alpha_{n-1})$, and:

i) for $i < n$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j)$;

ii) for $i = n$, $T[\alpha_n, l](A_i, j) \neq T[\alpha_{n-1}, 0](A_k, j)$;

(c) if $A_n = A_k \rightarrow A_e$, $k < n$, $e < n$, $J^l(\alpha_n) = J^0(\alpha_{n-1})$, $\tau_n = l = 0$, and:

i) for $i < n$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j)$;

ii) for $i = n$, $T[\alpha_n, l](A_i, j) = 1$ iff $T[\alpha_{n-1}, 0](A_k, j) = 0$ or $T[\alpha_{n-1}, 0](A_e, j) = 1$;

(d) if $A_n = \Box A_k$, $k < n$, then, for $\tau_n \geq l \geq 0$:

For every $j \in J^0(\alpha_{n-1})$,

I) let $\alpha(j, n\text{-}1) = \{j' \in J^0(\alpha_{n-1})$: $T[\alpha_{n-1}, 0](A_k, j') = 0$ and, for every r, $1 \leq r < n$, such that $A_r = \Box A_s$, $T[\alpha_{n-1}, 0](A_r, j) = T[\alpha_{n-1}, 0](A_r, j')\ \}$;

II) where $l < \tau_n(j)$, let $\gamma^l(j, n\text{-}1) = \{j' \in J^{l+1}(\alpha_n)$: $T[\alpha_n, l+1](A_k, j') = 0$ and, for every r, $1 \leq r < n$, if $A_r = \Box A_s$ and $T[\alpha_{n-1}, 0](A_r, j) = 1$, then $T[\alpha_n, l+1](A_r, j') = 1\}$;

III) for every p, every q, $q < p < n$ such that $A_p = \Box A_q$ and $T[\alpha_{n-1}, 0](A_p, j) = 0$, let $\beta(p, j, n\text{-}1) = \{j' \in J^0(\alpha_{n-1})$: $T[\alpha_{n-1}, 0](A_q, j') = 0$, $T[\alpha_{n-1}, 0](A_k, j') = 1$ and, for every r, $1 \leq r < n$, such that $A_r = \Box A_s$, $T[\alpha_{n-1}, 0](A_r, j) = T[\alpha_{n-1}, 0](A_s, j')\ \}$;

IV) for $l < \tau_n(j)$, for every p, every q, $q < p < n$ such that $A_p = \Box A_q$ and $T[\alpha_{n-1}](A_p, j, 0) = 0$, let $\delta^l(p, j, n\text{-}1) = \{j' \in J^{l+1}(\alpha_n)$: $T[\alpha_n, l+1](A_q, j') = 0$, $T[\alpha_n, l+1](A_k, j') = 1$ and, for every r, $1 \leq r \leq n$, if $A_r = \Box A_s$ and $T[\alpha_{n-1}, 0](A_r, j) = 1$, then $T[\alpha_n, l+1](A_r, j') = 1\}$;

(α) Let us take the case where $l = \tau_n$. For every $j \in J^0(\alpha_{n-1})$ such that $\tau_n(j) = l$,

V) let $\{j_1,\ldots j_m\} \subseteq J^0(\alpha_{n-1})$ such that $\tau_n(j) = l$ and:

1) $j_{m'} < j_{m''}$ if $m' < m''$;

2) $j_{m'} \in \{j_1,\ldots j_m\}$ if $\alpha(j_{m'}, n\text{-}1) \neq \emptyset$, for every $p < n$, $\beta(p, j_{m'}, n\text{-}1) \neq \emptyset$, and $T[\alpha_{n-1}, 0](A_k, j) = 1$.

Then $J^l(\alpha_n) = \{1,\ldots,q,\ldots,q+m\}$ and:

i) for $i < n$, $j \leq q$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j)$;

ii) for $i < n$, $j = q + m'$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j_{m'})$;

iii) for $i = n$, j such that $\alpha(j, n\text{-}1) = \emptyset$, $T[\alpha_n, l](A_i, j) = 1$;

iv) for $i = n$, j such that $\alpha(j, n\text{-}1) \neq \emptyset$ and, for some $p < n$, $\beta(p, j, n\text{-}1) = \emptyset$, $T[\alpha_n, l](A_i, j) = 0$;

v) for $i = n$, $T[\alpha_{n-1}, 0](A_k, j) = 0$, $T[\alpha_n, l](A_i, j) = 0$;

vi) for $i = n$, j such that $\alpha(j, n\text{-}1) \neq \emptyset$, for every $p < n$, $\beta(p, j, n\text{-}1) \neq \emptyset$, and $T[\alpha_{n-1}, 0](A_k, j) = 1$, in which case, for some $m' \in \{1,\ldots,m\}$, $j = j_{m'}$ or $j = q + m'$,

1) if $j = j_{m'}$ then $T[\alpha_n, l](A_i, j) = 1$;

2) if $j = q + m'$ then $T[\alpha_n, l](A_i, j) = 0$;

Now, for those $j \in J^0(\alpha_{n-1})$ such that $\tau_n(j) < l$,

vii) for $i < n$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, 0](A_i, j)$;

viii) for $i = n$, $j = q + m'$, $T[\alpha_n, l](A_i, j) = *$;

(β) Let us now take the case where $l < \tau_n$. For every $j \in J^0(\alpha_{n-1})$ such that $\tau_n(j) = l$,

VI) let $\{j_1,\ldots j_m\} \subseteq J^0(\alpha_{n-1})$ such that:

1) $j_{m'} < j_{m''}$ if $m',< m''$;

2) $j_{m'} \in \{j_1,\ldots j_m\}$ if $\alpha(j_{m'}, n\text{-}1) \neq \emptyset$ or $\alpha^l(j_{m'}, n\text{-}1) \neq \emptyset$; for every $p < n$, $\beta(p, j_{m'}, n\text{-}1) \neq \emptyset$ or $\beta^l(p, j_{m'}, n\text{-}1) \neq \emptyset$; and $T[\alpha_{n-1}, 0](A_k, j) = 1$.

Then $J(\alpha_n) = \{1,\ldots,q,\ldots,q+m\}$ and:

i) for $i < n$, $j \leq q$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, l+1](A_i, j)$;

ii) for $i < n$, $j = q + m'$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n-1}, l+1](A_i, j_{m'})$;

iii) for $i = n$, j such that $\alpha(j, n\text{-}1) = \emptyset$ and $\gamma^l(j, n\text{-}1) = \emptyset$, $T[\alpha_n, l](A_i, j) = 1$;

iv) for $i = n$, j such that $\alpha(j, n\text{-}1) \neq \emptyset$ or $\gamma^l(\text{j, n-1}) \neq \emptyset$, and, for some $p < n$, $\beta(p, j, n\text{-}1) = \emptyset$ and $\delta^l(p, j, n\text{-}1) = \emptyset$, $T[\alpha_n, l](A_i, j) = 0$;

v) for $i = n$, $T[\alpha_{n-1}, 0](A_k, j) = 0$, $T[\alpha_n, l](A_i, j) = 0$;

vi) for $i = n, j$ such that $\alpha(j, n\text{-}1) \neq \emptyset$ or $\gamma^l(j, n\text{-}1) \neq \emptyset$; for every $p < n$, $\beta(p, j, n\text{-}1) \neq \emptyset$ or $\delta^l(p, j, n\text{-}1) \neq \emptyset$; and $T[\alpha_{n\text{-}1}, 0](A_k, j) = 1$, in which case, for some $m' \in \{1,...,m\}$, $j = j_{m'}$ or $j = q + m'$,

1) if $j = j_{m'}$ then $T[\alpha_n, l](A_i, j) = 1$;

2) if $j = q + m'$ then $T[\alpha_n, l](A_i, j) = 0$;

Now, for those $j \in J^0(\alpha_{n\text{-}1})$ such that $\tau_n(j) < l$,

vii) for $i < n$, $T[\alpha_n, l](A_i, j) = T[\alpha_{n\text{-}1}, 0](A_i, j)$;

viii) for $i = n, j = q + m'$, $T[\alpha_n, l](A_i, j) = *$;

And for $j \in J^0(\alpha_{n\text{-}1})$ such that $\tau_n(j) > l$,

ix) for $i \leq n$, $T[\alpha_n, l](A_i, j) = T[\alpha_n, l+1](A_i, j)$.

So the main idea of the definition is to first extend to A_n the lines which have a maximum number of true necessities, and then proceed stepwise until we reach the "top" lines—the ones which have a minimum number of true necessities. It is also to notice that we introduced a third, dummy truth-value. This is so just in order to have the GTT function giving a value to every line on every step.

The next steps are now quite straightforward. We first prove that lines in a GTT which extends another preserve the values the wffs get:

Lemma L23. *Let α_n be a normal sequence. For every $j \in J^0(\alpha_k)$, $1 \leq k \leq n$, there is $j^* \in J^0(\alpha_n)$ such that, for every i, $1 \leq i \leq k$, $T[\alpha_k, 0](A_i, j) = T[\alpha_n, 0](A_i, j^*)$.*

Proof. This lemma can be easily proved by induction on $n\text{–}k$, based on conditions i) and ii) of D16.1.a through D16.1.d.

The first important result here is to show that saturated sets "coincide" with lines of a GTT, giving some finite set of wffs.

Lemma L24. *Let α_n be a normal sequence. For every saturated set Δ there is a $j \in J^0(\alpha_n)$ such that, for $1 \leq i \leq n$, $A_i \in \Delta$ iff $T[\alpha_n, 0](A_i, j) = 1$.*

Proof. Let Δ be a saturated set, and let $A_1,...,A_m$ (i.e., α_m) be the longest initial segment of α_n in which no wff occurs whose main operator is a necessity. We make first an induction on m.

(1) $m = 1$. By D16, $J^0(\alpha_1) = \{1,2\}$. If $A_i \notin \Delta$, from D16.1 there is $j = 2$ such that $T[\alpha_m, 0](A_i, j) = 0$; and, if $A_i \in \Delta$, from D16.1 there is $j = 1$ such that $T[\alpha_m, 0](A_i, j) = 1$.

(2) Let $m > 1$, and $J^0(\alpha_{n\text{-}1}) = \{1,...,q\}$. Induction hypothesis: for every saturated set Δ there is a $j \in J^0(\alpha_{m\text{-}1})$ such that, for $1 \leq i < j$, $A_i \in \Delta$ *iff* $T[\alpha_{m\text{-}1}, 0](A_i, j) = 1$. Now we have in this segment only propositional formulas, and the cases are where A_i is a variable, a negation or an implication, which are treated in a similar way as in the proof of Lemma L21. Thus we conclude

(†) for every saturated set Δ there is a $j \in J^0(\alpha_m)$ such that, for $1 \leq i \leq m$, $A_i \in \Delta$ *iff* $T[\alpha_m, 0](A_i, j) = 1$.

We now proceed with the main proof of this lemma, which goes by induction on n. For $n = 1,...,m$, the lemma is proved, so let us consider $n > m$.

(A) $n = m+1$. Here we have the first occurrence of a modalized wff, that is, for some $k < n$, $A_n = \Box A_k$. Obviously enough, $\tau_n = 0$.

(I) Suppose $A_n \notin \Delta$. By P22 (which also holds for KT4) we have an A_k-saturated set Θ such that $\varepsilon(\Delta^{\Box}) \subseteq \Theta$. From (†) there is a line $j \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$; and also that there is a line $j' \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Theta$ *iff* $T[\alpha_{n-1}, 0](A_i, j') = 1$. That is, $T[\alpha_{n-1}, 0](A_k, j) = 0$. Thus $\alpha(j, n-1) \neq \emptyset$. By D16.2.d.α.iv (or vi), $T[\alpha_n, 0](A_n, j) = 0$.

(II) Suppose $A_n \in \Delta$. From (†) there is a line $j \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$. All we have to show (since there are no other modalities) is that $T[\alpha_n, 0](A_n, j) = 1$. Since $k < n$, and since $A_k \in \Delta$, $T[\alpha_n\ 0](A_k, j) = 1$; then it follows from D16.2.d.α.vi.1, that $T[\alpha_n, 0](A_n, j) = 1$.

(B) $n > m+1$. Having thus proven the lemma for the base case of the first occurrence of a modal formula, we arrive to the following first inductive hypothesis:

(IH1) for every saturated set Δ there is a $j \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$.

The cases in which A_n is a propositional variable, a negation or an implication are proved as usual. Let us again consider the case in which, for some $k < n$, $A_n = \Box A_k$. Suppose also that $\tau_n \neq 0$. Now let l denote the cardinality of $\{\alpha_{n-1}\}^{\Box} \cap \Delta$. We proceed by making a new induction on l.

(α) $l = \tau_n$.

(I) Suppose $A_n \notin \Delta$. By P22 (which also holds for KT4) we have an A_k-saturated set Θ such that $\varepsilon(\Delta^{\Box}) \subseteq \Theta$. From L14.e.ii, $\Delta^{\Box} \subseteq \varepsilon(\Delta^{\Box})$; thus $\Delta^{\Box} \subseteq \Theta$. From (†) there is a line $j \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$; and also that there is a line $j' \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Theta$ *iff* $T[\alpha_{n-1}, 0](A_i, j') = 1$. Then we have that, for every $A_r = \Box A_s$, such that $A_r \in \Delta$, $T[\alpha_{n-1}, 0](A_r, j') = 1$. Now, since $l = \tau_n$—i.e., l is maximum—$\{\alpha_{n-1}\}^{\Box} \cap \Delta = \{\alpha_{n-1}\}^{\Box} \cap \Theta$. Thus $T[\alpha_{n-1}, 0](A_r, j) = T[\alpha_{n-1}, 0](A_r, j')$. Moreover, $T[\alpha_{n-1}, 0](A_k, j') = 0$, since Θ is A_k-saturated. This means that $\alpha(j, n-1) \neq \emptyset$, hence, by D16.2.d.α.iv (or vi), $T[\alpha_n, l](A_n, j) = 0$.

(II) Suppose $A_n \in \Delta$. From (†) there is a line $j \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$. Since $k < n$, and since $A_k \in \Delta$, $T[\alpha_n\ 0](A_k, j) = 1$. Since l is maximum, there are no necessities in $\{\alpha_{n-1}\}^{\Box}$ which doesn't belong to Δ, so $T[\alpha_n, l](A_n, j') = 1$ and we are done with the base case.

(β) $l < \tau_n$. We have the second inductive hypothesis:

(IH2) for every saturated set Δ, if the cardinality of $\{\alpha_{n-1}\}^{\Box} \cap \Delta$ is greater then l, there is a $j \in J^{l+1}(\alpha_n)$ such that, for $1 \leq i \leq n$, $A_i \in \Delta$ *iff* $T[\alpha_n, l+1](A_i, j) = 1$.

(I) Suppose $A_n \notin \Delta$. By P22 (which also holds for KT4) we have an A_k-saturated set Θ such that $\varepsilon(\Delta^{\Box}) \subseteq \Theta$. From L14.e.ii, $\Delta^{\Box} \subseteq \varepsilon(\Delta^{\Box})$; thus $\Delta^{\Box} \subseteq \Theta$. From (†) there is a line $j \in J^0(\alpha_{n-1})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$. Now, if the cardinality of $\{\alpha_{n-1}\}^{\Box} \cap \Theta$ is equal to l, the proof goes as in (α.1). So suppose it is bigger than l. From (IH2) there is a line $j' \in J^{l+1}(\alpha_n)$ such that, for $1 \leq i \leq n$, $A_i \in \Theta$ *iff* $T[\alpha_n, l+1](A_i, j') = 1$. Then we have that, for every $A_r = \Box A_s$, such that $A_r \in \Delta$, if $T[\alpha_{n-1}, 0](A_r, j) = 1$, then $T[\alpha_n, l+1](A_r, j') = 1$. Moreover, $T[\alpha_n, l+1](A_k, j') = 0$, since Θ is A_k-saturated. This means that $\gamma^l(j, n-1) \neq \emptyset$, hence, by D16.2.d.β.iv (or vi), $T[\alpha_n, l](A_n, j) = 0$.

(II) Suppose $A_n \in \Delta$. From (†) there is a line $j \in J^0(\alpha_{n-j})$ such that, for $1 \leq i < n$, $A_i \in \Delta$ *iff* $T[\alpha_{n-1}, 0](A_i, j) = 1$. Since $k < n$, and since $A_k \in \Delta$, $T[\alpha_n\ 0](A_k, j) = 1$. Suppose now there is some $A_p = \Box A_q$ such that $A_p \notin \Delta$. Similar to case (I), there is an A_q-saturated set Θ such that $\Delta^{\Box} \subseteq \Theta$. It is easy to see that $A_n \in \Theta$. A similar reasoning as in case (I)—now considering β and δ^l will prove the lemma. ■

This lemma is important because, as we can easily prove (like in chapter 2 for EDLs, or chapter 3 for normal modal logics) that a wff A is a theorem of KT4 if and only if it belongs to every saturated set. We are not going to prove it here, but will make use of this.

Lemma L25. *If* $\vdash A$ *then, for every normal sequence* α_n *where, for some* i, $A = A_i$, *and for every* $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A_i, j) = 1$.

Proof. By induction on the number r of lines of a proof of A_i in **KT4**.

(1) $r = 1$. The cases in which A_i is an axiom of **PL** are straightforward. Let us consider the other cases.

(a) Let $A_i = \Box(A \rightarrow B) \rightarrow (\Box A \rightarrow \Box B)$. Let us suppose that, for some $j \in J^0(\alpha_\eta)$, $T[\alpha_n, 0](A_i, j) = 0$. By D16.2.c.ii, $T[\alpha_n, 0](\Box(A \rightarrow B), j) = T[\alpha_n, 0](\Box A, j) = 1$ and $T[\alpha_n, 0](\Box B, j) = 0$. By D16.2.d we see that, since $T[\alpha_n, 0](\Box B, j) = 0$, that $\alpha(j, n-1) \neq \emptyset$ or $\gamma^{j}(j, n-1) \neq \emptyset$ (if not, by D16.2.d.β.iii, we would have $T[\alpha_n, 0](\Box B, j) = 1$). From this fact it follows that there is $j_n \in J^0(\alpha_n)$ such that $T[\alpha_n, 0](B, j_n) = 0$ and, for every r, $1 \leq r < n$ such that $A_r = \Box A_s$ and $T[\alpha_n, 0](A_r, j) = 1$, $T[\alpha_n](Ar_s, j_n) = 1$. Well, $T[\alpha_n, 0](\Box(A \rightarrow B), j) = T[\alpha_n, 0](\Box A, j) = 1$, thus $T[\alpha_n, 0](A \rightarrow B, j_n) = T[\alpha_n, 0](A, j_n) = 1$ (else, by D16.2.d.β.v, $T[\alpha_n, 0](\Box(A \rightarrow B), j) = T[\alpha_n, 0](\Box A, j) = 0$). It thus cannot be that $T[\alpha_n, 0](B, j_n) = 0$. Thus for every $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A, j) = 1$.

(b) Let $A_i = \Box A \rightarrow A$. Let us suppose that, for some $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A_i, j) = 0$. But then $T[\alpha_n, 0](\Box A, j) = 1$ and $T[\alpha_n, 0](A, j) = 0$—against D16.2.d.β.v.

(c) Let $A_i = \Box A \rightarrow \Box\Box A$. Let us suppose that, for some $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A_i, j) = 0$. By D16.2.c.ii, $T[\alpha_n, 0](\Box A, j) = 1$ and $T[\alpha_n, 0](\Box\Box A, j) = 0$. By D16.2.d we see that, since $T[\alpha_n, 0](\Box\Box A, j) = 0$, that $\alpha(j, n-1) \neq \emptyset$ or $\gamma^{j}(j, n-1) \neq \emptyset$ (if not, by D16.2.d.β.iii, we would have $T[\alpha_n, 0](\Box\Box A, j) = 1$). However, in both cases we'll supposing there is a line j' of the table in which $\Box$A has value 0 (since $T[\alpha_n, 0](\Box\Box A, j) = 0$), but which also has to give value 1 to this formula (since $T[\alpha_n, 0](\Box A, j) = 1$ and, by definition of $\alpha(j, n-1)$ and $\gamma^{j}(j, n-1)$, they satisfy the necessities of j). Since this is impossible, it cannot be that $T[\alpha_n, 0](A_i, j_n) = 0$. Thus for every $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A, j) = 1$.

(2) $r > 1$. In this case, either A_i is an axiom, and the property is already proved in 1), or it was obtained by one of the inference rules *MP* or *RN*, in which case the proof is analogous to the case of **K** in the preceding chapter. ■

Theorem T29. $\vdash A$ *iff for every normal sequence* α_n *where, for some* i, $A = A_i$, *and for every* $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A_i, j) = 1$.

Proof. One direction is the preceding lemma, so suppose that, for every normal sequence α_n where, for some i, $A = A_i$, and for every $j \in J^0(\alpha_n)$, $T[\alpha_n, 0](A_i, j) = 1$. Suppose furher that $\nvdash$ A. It is easy to prove that, for some saturated set Δ, A $\notin$ Δ. By L25 there is a $j \in J^0(\alpha_n)$ such that $T[\alpha_n, 0](A_i, j) = 0$—against our hypothesis. Thus $\vdash$A. ■

As we just saw, we have then used sucessfully GTTs for S4. A similar technique can be used then with the other normal modal logics like **K5**.

7

Valuations, possible worlds, and tableau systems

Non ovum tam simili ovo,
quam hic illi est.

I guess it would be interesting, now that we just finished our journey through the (alethic modal) Valuation Semantics Jungle, to say a few words comparing this semantics to other ones. Thus, in this small chapter, we'll give some thought to two questions, namely: what is the relation (if any) between valuation semantics and the ordinary Kripke or possible-world semantics, on the one side, and, on the other, what is the relation between GTTs and tableau systems.

7.1 Valuation and Possible World Semantics

I already mentioned in chapter 4, while introducing the subject, that there is some kind of relation between valuation and possible world semantics. If we want, we can certainly see a valuation as a world, or as a function describing a world. Remember, we have proven that valuations are characteristic functions of maximal consistent sets, and what is an MCS, one could ask, if not a world? From this point of view, the only model we have to consider is the class of all MCSs—we don't need anymore to introduce acessibility relations as primitive elements of the model. To be precise, in the first chapter of this work, where we were discussing semantics for EDLs, we came exactly across this fact: that the class of all saturated sets (which are maximal consistent), and the class of sets of wffs true in some world of some (EDL) model, were the same. For the sake of completeness, let us write everything down here, taking a normal modal logic as an example, say **KDB**.

The first step is to define a possible-world model for **KDB**. As usual, it is a structure $<W, R>$, where W is a non-empty set (of worlds) and R a binary relation over W which has the properties of seriality and symmetry. To this structure we add an interpretation function I which assign a truth-value 1 or 0 to the atomic formulas. In the usual way we extend I to a function from $W \times \mathbf{FOR}_{\mathbf{KDB}}$ into $\{0, 1\}$, such that, for some $w \in W$:

$I(\neg A, w) = 1$ iff $I(A, w) = 0$;
$I(A \rightarrow B, w) = 1$ iff $I(A, w) = 0$ or $I(B, w) = 1$;
$I(\Box A, w) = 1$ iff for every $v \in W$ such that wRv, $I(A, v) = 1$.

Now this tuple $<W, R, I>$ is a possible-world model for **KDB**. I am not going to prove that it is: the interested reader can consult, say, [Ch80], or any good textbook on modal logic.

Now, in a similar fashion as we did in Chapter 1 with respect to EDLs, we define an equivalence relation among **KDB**–models. Let **K** be the set of all **KDB**–models. If $\mathcal{M}$ and $\mathcal{N} \in$ **K**, we say that $\mathcal{M} \approx \mathcal{N}$ in case $\mathcal{M} \models A$ iff $\mathcal{N} \models A$, for every wff A. If $\mathcal{M}$ is a model, $|\mathcal{M}|$ will denote equivalence class of $\mathcal{M}$ (thus $|\mathcal{M}| \in \mathbf{K}_{/\approx}$). Let now $\mathcal{M} = <W, R, I>$ be a model. For each $w \in W$, let $[\mathcal{M}, w] = \{A: I(A,w) = 1\}$. Let $\mathbf{W} = \{\Gamma \subseteq \mathbf{FOR} : \Gamma = [\mathcal{M}, w]$, for some $\mathcal{M}$, some $w\}$. And finally, let $\mathbf{S}$ be the class of all **KDB**–saturated sets.

We can now prove the following results:

Lemma L26. *If* $[\mathcal{M}, w] \vdash A$ *then* $I(A,w) = 1$.

Proof. Analogous to the proof of L2.

Lemma L27. $\mathbf{W} = \mathbf{S}$.

Proof. Similar the proof of L3, or, by that matter, of L7.

Theorem T30. *There is a bijective function h from the set V of all valuations into the set* $\mathbf{K}_{/\approx}$ *such that if* $v \in V$, $h(v) = |\mathcal{M}|$, *and if* $\mathcal{M} = <W,R,I>$ *then for every* $A \in \mathbf{FOR}_{\mathbf{KDB}}$, $v(A) = I(A,w)$.

Proof. Follows from the Corollary to T21, and from L3.[45] ■

So this is the way in which valuation and possible-world semantics are related. But I would like to stress again that, in spite of the similarities, they are not the same thing, specially because it is the inductive definition of an $A_1,...,A_n$–valuation, for some normal sequence $A_1,...,A_n$, which allows us to easily obtain a decision procedure via GTTs.

7.2 GTTs and Tableau Systems

It is very likely that the first thing that comes to your mind, if you are familiar with tableau-style theorem provers for nonclassical logics, is the question whether GTTs are just the same as these. What they aren't, as little as, in classical propositional logic, truth-tables are the same thing as tableau systems. But of course they are related. To give a short answer—I'm going to explain it later on—they tackle the decision problem from opposite ends.

[45] Cf. [Lo77], p. 152, where this theorem is proven in the case of **K**.

The history of tableau systems (cf. [Fi83], pp. 3-10) could be said to begin in 1935, with the introduction by Gerhard Gentzen of the proof systems nowadays known as "Gentzen's sequent calculus". The nice feature of these proof systems (once you've proven that a certain inference rule named *cut* can be disposed of without losses) is that they obeyed the so-called *subformula property*: that is, in a proof of some formula *A* we only need to consider subformulas of *A*. That this is an awfully nice property should be clear to everyone who spent some time trying to find proof of theorems in a "get it from the axioms" way.

Gentzen's work was developed first by Beth, and later by Smullyan (for classical logic). The result were "upside down" Gentzen type systems, which consists in what we now call tableau systems ([Fi83], p. 5). In the modal logic case, we have the contributions of Hintikka, Kripke, Hughes and Cresswell, and, of course, Fitting himself.[46]

Now the main feature of tableau proof procedures is that they are *refutation* systems. That is, one tries to generate a countermodel for the formula in question: the formula is assumed to be false, and one proceeds by computing which value its subformulas would have under this assumption. Since at each step we reduce a formula to its subformulas (which are smaller), and since formulas have a finite length, the method is sure to terminate: somewhen there are no more subformulas to be processed, and atoms, of course, cannot be further reduced.

The whole construction is made in an inverted-tree manner: we write at the top (of a sheet of paper, for instance) the formula *A* to be (dis)proven, preceded by a *sign*: T or F, which informs us whether the formula is true or false. Since the first wff is the one we are trying to refute, it gets an F. Now we proceed way down on the paper by adding new nodes to this seed of a tree: at each node there is going to be a signed subformula of *A*. In other words, we enlarge the tree using a set of *extension rules* ([Fi83], p. 29).

In this adding of subformulas to the tree we can distinguish two cases. First, sometimes there is only one possible way of assigning values to subformulas, like when we have a true conjunction $A \wedge B$: *both* conjuncts must be true, if their conjunction is; so we extend the tree by adding both T*A* and T*B*. Sometimes, however, we are confronted with two possibilities: with a true implication $A \rightarrow B$, for instance, one has that either the antecedent is false, or the consequent is true. In order to account for these two possibilities, the branch we are working on must be split into two new branches, each of them representing a way of going on (a possible assignment). Branches can of course split further into sub-branches, and sub-sub-branches, and this is why the tableau ends up being a tree.

For the classical propositional logic, the tableau extension rules are the following (cf. [Fi83], pp. 29-30):

$$\mathrm{T}\wedge:\quad \frac{\mathrm{T}(A\wedge B)}{\begin{array}{c}\mathrm{T}A\\ \mathrm{T}B\end{array}}\qquad\qquad \mathrm{F}\wedge:\quad \frac{\mathrm{F}(A\wedge B)}{\mathrm{F}A \quad\mid\quad \mathrm{F}B}$$

$$\mathrm{T}\vee:\quad \frac{\mathrm{T}(A\vee B)}{\mathrm{T}A \quad\mid\quad \mathrm{T}B}\qquad\qquad \mathrm{F}\vee:\quad \frac{\mathrm{F}(A\vee B)}{\begin{array}{c}\mathrm{F}A\\ \mathrm{F}B\end{array}}$$

[46] More historical details can be found in [Fi83].

$$\text{T}\to: \quad \begin{array}{c} \text{T}(A\to B) \\ \hline \text{F}A \;\mid\; \text{T}B \end{array} \qquad\qquad \text{F}\to: \quad \begin{array}{c} \text{F}(A\to B) \\ \hline \text{T}A \\ \text{F}B \end{array}$$

$$\text{T}\neg: \quad \begin{array}{c} \text{T}\neg A \\ \hline \text{F}A \end{array} \qquad\qquad \text{F}\neg: \quad \begin{array}{c} \text{F}\neg A \\ \hline \text{T}A \end{array}$$

After having applied the extension rules, we find that, in the end, two things can occur:

(1) We find that every branch leads to a contradiction, i.e., for some atom p, Tp and Fp belong to the branch. In this case, the branch is said to be *closed*. All branches being closed, the supposition that the original formula could be false is absurd, hence it must be valid.

(2) Some branch remains open, i.e., there is no more complex formulas in the branch which we have still not processed, and no contradiction arose. In this case, what we have done amounts to actually *creating* a model which falsifies our formula—hence it is not valid.

Since a picture is better than ten thousand words, let us look at a tableau for $A = (a\to b)\to(\neg b\to\neg a)$:

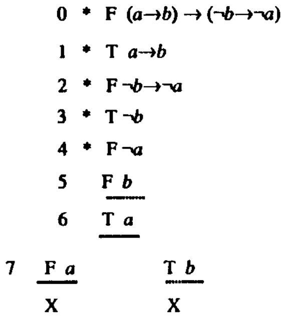

fig. 21

We began by writing down 'FA'—this is what line 0 means. It is a false implication, so its antecedent must be true and its consequent false; hence we add both of them, with the corresponding signs, and cross FA out (we put a star in front of it to show it was already reduced). Now we have two new formulas to which we can apply the rules: T$a\to b$ and F$\neg b\to\neg a$. Since the first of these would entail a branching in the tableau, we reduce first the second one, that is how we obtain T$\neg b$ and F$\neg a$ on lines 3 and 4. This is the First Very Important Tableau Rule: if you can avoid branching, then do it, else you'll be complicating things unnecessarily. After further reducing we get the atoms Fb and Ta (lines 5 and 6). Now we come to the point where there are no more non-branching formulas, so we work on T$a\to b$. An implication is true either if its antecedent is false, or its consequent is true: these two possibilities are represented by branching the tableau in line 7. Each node begins a possible continuation. Now in each branch we find a contradiction (the underlined atoms in the tree), so both branches are closed (denoted by the 'X' at the bottom). Since every possible way of assigning values to the propositional variables of *A* led to an absurd, it must be a tautology. And it is.

Summing it ali up, we say that a *tableau* is *closed* iff every one of its branches is closed. And a closed tableau for a wff *A* is then said to be a *proof* of *A*. (Cf. [Fi83], p. 30)

Now, how are we going to extend this construction to modal logics? An answer is to be found in, for instance, [HC72], or, more complete, in [Fi83]. I'm going to take here **K** as an example, since we are wanting to make a small comparison to GTTs, and we did that for **K** in a previous chapter. What we need are, obviously enough, extension rules for the cases in which we want to reduce some modalized formula. For this we could use some intuitions from the semantics. Drawing again on metaphors, we can say that a tableau (for **PL**) is (part of) a world. In a modal logic, to evaluate a formula we have sometimes to consider formulas in other worlds as well. So the answer should be something along the following line: when you find a modalized formula, say F□*A*, create another, alternate tableau (Cf. [Fi83], p. 34). For the general case, as Fiting points out, the difficulty with this idea is that "in practice it gets rather messy keeping track of the alternatives in a tableau proof of even a moderately complex formula" ([Fi83], p. 34). But for some systems, **K** included, that will do nicely. Now one just has to define what to carry on to the alternate tableau, when we create one.

Again, the semantics gives the answer. Having found a F□*A*, of course we are going to create an alternate tableau ("an accessible world") in which we have F*A*. And since, if □*B* is true in a world, then *B* is true in ali accessible worlds, we have to add T*B* to the new tableau for every T□*B* we had in the older one, the case of (im)possibilities being handled in a similar, mirroring way. Thus, if S is the set of formulas in a branch, we define $S\# = \{TA : T\Box A \in S\} \cup \{FA : F\Diamond A \in S\}$. Then we have the following two rules (in **K** we don't have rules for F◇ and T□):

$$\text{T}\Diamond: \quad \frac{S,\ \text{T}\Diamond A}{S\#,\ \text{T}A} \qquad\qquad \text{F}\Box: \quad \frac{S,\ \text{F}\Box A}{S\#,\ \text{T}A}$$

An example:

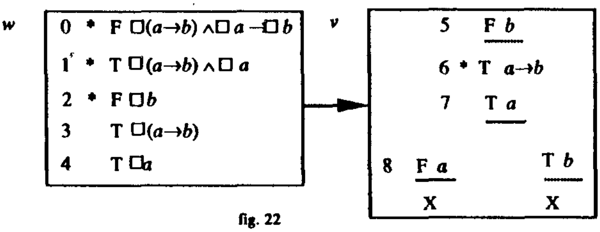

fig. 22

In the picture we see that we could work on tableau *w* until we were only left with modalized formulas (lines 2–4). Since K has no rules for F◇ and T□, our only possibility was to process line 3. We thus opened a new tableau *v*, into which we carried F*b* and the scopes of the true necessities in *w*: T*a*→*b* and Ta. With these three lines we were then able to find a contradiction in v, closing both branches. Thus the original wff is valid in **K**.

One can now show that the tableau proof procedure just described for **K** in fact works. That is, a wff A is valid in **K** iff there is a closed tableau for A. (I'm not going to do this here, because it is a standard result; the reader can consult e.g. [Fi83], chapter 2.) What is of concern to us here is the fact that a branch of a tableau is satisfiable iff there is a world w in a Kripke model $\mathcal{M}$ such that all wffs on the branch get are true in w. Since a world in a Kripke model corresponds to a saturated set (cf. L28 above), and thus to a valuation (Corollary to T21), and thus (by L21) to a line of a generalized truth-table, we conclude that a branch of a tableau is satisfiable iff there is a line on some GTT which gives 1 to every formula on the branch.

What is then the difference? As I put it before, these methods tackle the decision problem from opposite ends. Whereas tableau systems are refutation procedures—one tries to build a countermodel, reaching a contradiction if none exists—GTTs try to examine all relevant models, and see whether the formula is true in ali of them. This is the same situation which obtains between truth-tables and tableau systems for classical propositional logic. As I said in the introduction, they are the two sides of a coin.[47]

[47] There is still another way of doing tableau systems, which is somewhat different from Fitting's formulation: I mean the way G.E. Hughes and M.J. Cresswell present semantical tableaus in their [HC72]. Their formulation is different from Fitting's in that they use a more graphical approach: worlds are represented through boxes containing formulas, and the accessibility relations holding among them are represented through arrows. Moreover, in systems like **KT5** or **KTB**, one goes back and forth between worlds, a feature that Fitting tries to eliminate because it increases the complexity of the computation. HC's formulation is what Fitting has in mind when he says (quotation above) that it is rather messy keeping track of a lot alternate tableaus.

In the case of valuation semantics, they have sometimes more similitudes to HC's way of doing things than to Fitting's, because, if we try to generate a tableau procedure out of a GTT (like we do in a next chapter on implementations) we'll see that we will also be going back and forth between "worlds", or between tableaus, using results from a "newer" one to derive a contradiciton in an "older" one.

8

Valuations & GTTs for Z5

Kinkler's Second Law:

All the easy problems have been solved.

After having developed in the preceding chapters the method of valuations for several systems of modal logic, now it has come the time to turn our attention again to epistemic-logical matters. In this chapter we are going to apply to EDLs what we have learned so far. I will choose here just one EDLogic as an example, and give for it the valuation definitions, GTT construction, and then proving that they are correct. After that we'll be ready to consider some implementation questions.

The logic I have chosen to take as an example is Z5, for the very simple reason that it is the system HM already mentioned. The strategy of this chapter is pretty much the same as in the modal logics case. Some things are of course going to stay the same—for instance, semi-valuations, and valuations simpliciter. Hence the main point is again finding a nice definition of an $A_1,\dots,A_n$-valuation for Z5.

8.1 Defining $A_1,\dots,A_n$-valuations for Z5

Before we get things rolling, it is worth mentioning that the situation here, on the one hand, is going to be more complicated than in the modal case, since we have two strong primitive operators ('B' and 'K'), in comparison to '□'. On the other hand, we won't be considering weak operators (like ◇), so things get in this aspect simpler. As a first consequence of this we don't need anymore the distinguishing subscripts in '$\models_1$' and '$\models_0$', because we'll be considering just the satisfaction case, and not rejection. That is, plain '$\models$' will be meaning our old '$\models_1$'.

Just to remember, we'll continue to use 'α' as a typographical substitution for '$A_1,\dots,A$', so 'α_{k-1}' means actually '$A_1,\dots,A_{k-1}$', and so forth.

But let us now begin by introducing the Z5 analog of $f{<}k{>}g$: we'll have here actually *two* analogs, since we have two strong operators (which has already given us, in the possible world models, the two accessiblitiy relations S and R). So let α_n be a normal sequence and f, g two functions from FOR into $\{0,1\}$. We say that, for $1 \leq k \leq n$,

(a) *(belief)* $f{<}\beta,k{>}\,g$ iff $g \vDash \varepsilon(\{\alpha_k\}^{\mathrm{B}}_{f,1})$,
$\{\alpha_k\}^{\mathrm{B}}_{f,1} = \{\alpha_k\}^{\mathrm{B}}_{g,1}$, and
$\{\alpha_k\}^{\mathrm{K}}_{f,1} = \{\alpha_k\}^{\mathrm{K}}_{g,1}$;

(b) *(knowledge)* $f{<}\kappa,k{>}\,g$ iff $\{\alpha_k\}^{\mathrm{K}}_{f,1} = \{\alpha_k\}^{\mathrm{K}}_{g,1}$.

A few words on this. The belief case is, firstly, similar to **KD45** one (with 'B' instead of '□')—that's what the first two clauses say (cf. Abb2 on chapter 4). Now the additional requirement that the true knowledge formulas of f and g must be the same is there in order to capture the idea that knowledge implies belief (or, in possible world semantics talking, that the belief accessibility relation is included in the knowledge one). The knowledge case, as one can see, is just plain **KT5** (with 'K' instead of '□').

We can now go immediately to the definition of an α_n–valuation.

Definition D17. v is a α_n*–valuation* (for **Z5**) if α_n is a normal sequence and:
1) $n = 1$ and v is a semi-valuation;
2) $n > 1$, v is an α_{n-1}–valuation and, if for some $m < n$,
A) $A_n = \mathrm{B}A_m$,
I) if $v(A_n) = 0$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v{<}\beta, n\text{-}1{>}v_n$;
II) if $v(A_n) = 1$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 1$ and $v{<}\beta, n\text{-}1{>}v_n$; moreover, for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$, $v_p(A_m) = 1$ and $v{<}\beta, n\text{-}1{>}v_p$.
B) $A_n = \mathrm{K}A_m$,
I) if $v(A_n) = 0$ then there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v{<}\kappa, n\text{-}1{>}v_n$;
II) if $v(A_n) = 1$ then $v(A_m) = 1$ and for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{K}A_q$ [$A_p = \mathrm{B}A_q$] and $v(A_p) = 0$ there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$, $v_p(A_m) = 1$ and $v{<}\kappa, n\text{-}1{>}v_p$ [$v{<}\beta, n\text{-}1{>}v_p$].

As one can see, precious little has changed from the corresponding definitions for the normal modal logics—we just needed some minor adjustments. It is however worth noticing that, in the knowledge case, false belief formulas (the requirement "… [$A_p = \mathrm{B}A_q$] and $v(A_p) = 0$…") must also be taken into account (they entail that the corresponding knowledge wff will also be false). Having now defined α_n–valuations, the rest follows as usual. First we make the necessary changes in the definition of a *canonical extension*:

Definition D18. Let α_n be a normal sequence and v an α_{n-1}–valuation. We say that v_c is the *canonical extension* of v to α_n if:

A) for ali $m < n$, $A_n \neq \mathrm{B}A_m$, $A_n \neq \mathrm{K}A_m$ and $v_c = v$; or

B) for some $m < n$, $A_n = \mathrm{B}A_m$ [$A_n = \mathrm{K}A_m$] and v_c is a function from FOR into $\{0,1\}$ such that, for every formula B,

1) if A_n is not a subformula of B, then $v_c(B) = v(B)$;

2) if A_n is a subformula of B, then

a) for $B = A_n$, $v_c(B) = 0$ iff there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$ and $v<\beta, n\text{-}1>v^+$ [$v<\kappa, n\text{-}1>v^+$];

b) for $B = \neg C$, $v_c(B) = 1$ iff $v_c(C) = 0$;

c) for $B = C \rightarrow D$, $v_c(B) = 1$ iff $v_c(C) = 0$ or $v_c(D) = 1$;

d) for $B = \mathrm{B}C$ or $B = \mathrm{K}C$, $v_c(B) = v(B)$.

We only need now the definitions of *normality*, and then things can get going. Let v be an α_n–valuation: for $1 \leq k \leq n$, we say that v is B$_0$–α_k–*normal* [K$_0$–α_k–*normal*] if for every p, every q, $q < p \leq k$, such that $A_p = \mathrm{B}A_q$ [$= \mathrm{K}A_q$] and $v(A_p) = 0$, there is an α_k–valuation v_p such that $v_p(A_q) = 0$ and $v<\beta, k>v_p$ [$v<\kappa, k>v_p$]. We say that a valuation v is B$_1$–α_k–*normal* if for every p, every q, $q < p \leq k$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 1$, there is an α_k–valuation v_p such that $v_p(A_q) = 1$ and $v<\beta, k>v_p$; v is K$_1$–α_k–*normal* if for every p, every q, $q < p \leq k$, such that $A_p = \mathrm{K}A_q$ and $v(A_p) = 1$, $v(A_q) = 1$. (B$_1$– and K$_1$–normality, remember, are the conditions which take care of axiom schemas k^b and k.)

We are now ready to get our results. That canonical extensions are semi-valuations should be by now obvious—it is proved exactly in the same way as in the normal modal logics case, so nothing new here.

Proposition P26. *Let α_n be a normal sequence, v an α_{n-1}–valuation and v_c the canonical extension of v to α_n. Let us suppose that v is* B$_0$–, B$_1$–, K$_0$– *and* K$_1$–α_{n-1}*–normal. In this case, v_c is an α_n–valuation.*

Proof. First of all, v_c is an α_{n-1}–valuation, because it is a semi–valuation and, by construction, for $1 \leq i < n$, $v_c(A_i) = v(A_i)$. Now, if for every $m < n$, $A_n \neq \mathrm{B}A_m$, $A_n \neq \mathrm{K}A_m$, v_c fulfills every condition of D17, so it is an α_n–valuation. So suppose that, for some $m < n$, $A_n = \mathrm{B}A_m$ or $A_n = \mathrm{K}A_m$. We have two cases:

(I) $v_c(A_n) = 0$. By D18.B.2.a there is an α_{n-1}–valuation v^+ such that $v^+(A_m) = 0$ and $v<\beta, n\text{-}1>v^+$, or $v<\kappa, n\text{-}1>v^+$. Since v and v_c agree for $i < n$, $v_c<\beta, n\text{-}1>v^+$, or $v_c<\kappa, n\text{-}1>v^+$. So v_c is an α_n–valuation.

(II) $v_c(A_n) = 1$. We get first that:

(†) By D18.B.2.a, for every α_{n-1}–valuation v^+ such that $v<\beta, n\text{-}1>v^+$ [or $v<\kappa, n\text{-}1>v^+$], $v^+(A_m) = 1$.

We consider now separately the belief and knowledge cases:

α) *Belief:*

Suppose there is $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $v_c(A_p) = 0$. Then $v(A_p) = 0$ and, since v is B$_0$–α_{n-1}–normal, there is an α_{n-1}–valuation v_p such that $v<\beta, n\text{-}1>v_p$ and $v_p(A_q) = 0$. Since v and v_c agree for $i < n$, we have that $v_c<\beta, n\text{-}1>v_p$, and, from (†), that $v_p(A_m) = 1$ (else we would have $v_c(A_n) = 0$). We have now to prove that there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 1$ and $v<\beta, n\text{-}1>v_n$. If there is some $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $v_c(A_p) = 0$, then we have already got this α_{n-1}–valuation v_p such that $v<\beta, n\text{-}1>v_p$ and $v_p(A_m) = 1$. Suppose then there is no $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $v_c(A_p) = 0$. We have two possibilities:

(i) there is some $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $v_c(A_p) = 1$. Then $v(A_p) = 1$ and, since v is B_1–$\alpha_{n\text{-}1}$–normal, there is an $\alpha_{n\text{-}1}$–valuation v_p such that $v{<}\beta, n\text{-}1{>}v_p$ and $v_p(A_q) = 1$. Since v and v_c agree for $i < n$, we have that $v{<}\beta, n\text{-}1{>}v_p$; it follows from (†) that $v_p(A_m) = 1$.

(ii) there is no $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $v_c(A_p) = 1$. Well, in this case, $\{\alpha_k\}^{\mathrm{B}} = \emptyset$, in which case $v_c \models \varepsilon(\{\alpha_k\}^{\mathrm{B}}{}_{v_c,1})$, and obviously enough $\{\alpha_k\}^{\mathrm{B}}{}_{v_c,1} = \{\alpha_k\}^{\mathrm{B}}{}_{v_c,1}$; $\{\alpha_k\}^{\mathrm{K}}{}_{v_c,1} = \{\alpha_k\}^{\mathrm{K}}{}_{v_c,1}$; so $v_c{<}\beta, n\text{-}1{>}v_c$ and, from (†), $v_c(A_m) = 1$.

It follows, in both cases, that v_c is an α_n–valuation.

β) *Knowledge:*

If there is $q < p \leq n$ such that $A_p = \mathrm{K}A_q$ and $v_c(A_p) = 0$, the proof goes as in α). If now there is $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $v_c(A_p) = 0$, then $v(A_p) = 0$ and, since v is B_0–$\alpha_{n\text{-}1}$–normal, there is an $\alpha_{n\text{-}1}$–valuation v_p such that $v{<}\beta, n\text{-}1{>}v_p$ and $v_p(A_q) = 0$. Since v and v_c agree for $i < n$, we have that $v_c{<}\beta, n\text{-}1{>}v_p$. Now this means, among other things, that $\{\alpha_k\}^{\mathrm{K}}{}_{v_c,1} = \{\alpha_k\}^{\mathrm{K}}{}_{v_p,1}$. That is, $v_c{<}\kappa, n\text{-}1{>}v_p$. From (†), we then get $v_p(A_m) = 1$ (else we would have $v_c(A_n) = 0$).

We have now to prove that $v_c(A_m) = 1$. Since, for every $\alpha_{n\text{-}1}$–valuation v^+ such that $v{<}\kappa, n\text{-}1{>}v^+$, $v^+(A_m) = 1$, we only need to prove that $v_c{<}\kappa, n\text{-}1{>}v_c$. But this is immediate, because $\{\alpha_k\}^{\mathrm{K}}{}_{v_c,1} = \{\alpha_k\}^{\mathrm{K}}{}_{v_c,1}$. Hence v_c is an α_n–valuation.■

Now to the next lemma, where we can show that α_n–valuations are normal without restrictions, and thus that they can be extended as long we we want them to be.

Lemma L26. *Let v be an α_n–valuation. Then v is* B_0–, B_1–, K_0– *and* K_1–*α_n–normal.*

Proof. By induction on n. For $n = 1$ it holds trivially, so let $n > 1$ and let us suppose that every $\alpha_{n\text{-}1}$–valuation is B_0–, B_1–, K_0– and K_1–$\alpha_{n\text{-}1}$–normal. It follows then from P26 that

(†) The canonical extensions of $\alpha_{n\text{-}1}$–valuations to α_n are α_n–valuations.

We have now our usual three cases:

(1) For every $m < n$, $A_n \neq \mathrm{B}A_m$, $A_n \neq \mathrm{K}A_m$. So v is trivially B_0–, B_1–, K_0– and K_1–α_n–normal.

(2) Let us suppose that, for some $m < n$, $A_n = \mathrm{B}A_m$.

(I) Let $v(A_n) = 0$. We have:

1) $\{\alpha_n\}^{\mathrm{B}}{}_{v,1} = \{\alpha_{n\text{-}1}\}^{\mathrm{B}}{}_{v,1}$;

2) $\{\alpha_n\}^{\mathrm{K}}{}_{v+,1} = \{\alpha_{n\text{-}1}\}^{\mathrm{K}}{}_{v+,1}$, for every $\alpha_{n\text{-}1}$–valuation v^+;

3) $\varepsilon(\{\alpha_n\}^{\mathrm{B}}{}_{v,1}) = \varepsilon(\{\alpha_{n\text{-}1}\}^{\mathrm{B}}{}_{v,1})$.

It follows that, for every $\alpha_{n\text{-}1}$–valuation v^+,

4) if $v{<}\beta, n\text{-}1{>}v^+$ then $v{<}\beta, n{>}v^+$.

From the induction hypothesis, v is B_0–$\alpha_{n\text{-}1}$–normal, so we have:

5) for every p, every q, $q < p < n$ sucht that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$, there is an $\alpha_{n\text{-}1}$–valuation v_p such that $v_p(A_q) = 0$ and $v{<}\beta, n\text{-}1{>}v_p$.

Now, for each p, let v_p^* be the canonical extension of v_p to α_n. Obviously $v_p^*(A_q) = v_p(A_q)$, and, from (†), v_p^* is an α_n–valuation. From this, 4) and 5):

6) for every p, every q, $q < p < n$ sucht that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p^* such that $v_p^*(A_q) = 0$ and $v<\beta, n>v_p^*$.

On the other hand, since v is an α_n–valuation, we have:

7) there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 0$ and $v<\beta, n-1>v_n$.

Now let v_n^* be the canonical extension of v_n to α_n. Obviously $v_n^*(A_m) = v_n(A_m)$, and, from (†), v_n^* is an α_n–valuation.

Thus we have from this fact, together with 4) and 7):

8) for $p = n$, $q = m$, $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p^* such that $v_p^*(A_q) = 0$ and $v<\beta, n>v_p^*$.

From 6) and 8), then, v is an B_0–α_n–normal.

Now, since $A_n = \mathrm{B}A_m$ and $v(A_n) = 0$, v is trivially B_1–, K_0–, and K_1–α_n–normal.

(II) Let $v(A_n) = 1$. We then have:

1) $\{\alpha_n\}^{\mathrm{B}}_{v,1} = \{\alpha_{n-1}\}^{\mathrm{B}}_{v,1} \cup \{A_n\}$;

2) $\{\alpha_n\}^{\mathrm{K}}_{v^+,1} = \{\alpha_{n-1}\}^{\mathrm{K}}_{v^+,1}$, for every α_{n-1}–valuation v^+;

3) $\varepsilon(\{\alpha_n\}^{\mathrm{B}}_{v,1}) = \varepsilon(\{\alpha_{n-1}\}^{\mathrm{B}}_{v,1}) \cup \{A_m\}$.

Since $v(A_n) = 1$, we have from definition 1 that:

4) for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$, $v_p(A_m) = 1$ and $v<\beta, n-1>v_p$.

For each p, let v_p^* be the canonical extension of v_p to α_n. Obviously $v_p^*(A_q) = v_p(A_q)$, and, from (†), v_p^* is an α_n–valuation. It follows that:

5) for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_n–valuation v_p^* such that $v_p^*(A_q) = 0$, $v_p^*(A_m) = 1$ and $v<\beta, n-1>v_p^*$.

We only need to prove now that $v<\beta, n>v_p^*$; the B_0–α_n–normality follows.

First, since $v_p^*(A_m) = 1$, $v_p^* \vDash_1 \varepsilon(\{\alpha_{n-1}\}^{\mathrm{B}}_{v,1}) \cup \{A_m\}$; thus, from 3),

6) $v_p^* \vDash_1 \varepsilon(\{\alpha_n\}^{\mathrm{B}}_{v,1})$.

We have now two cases to consider:

(A) If, now, $v_p^*(A_n) = 1$, $\{\alpha_n\}^{\mathrm{B}}_{v_p^*,1} = \{\alpha_{n-1}\}^{\mathrm{B}}_{v_p^*,1} \cup \{A_n\}$; $\{\alpha_n\}^{\mathrm{B}}_{v,1} = \{\alpha_n\}^{\mathrm{B}}_{v_p^*,1}$ and since (from 2) $\{\alpha_n\}^{\mathrm{K}}_{v_p^*,1} = \{\alpha_{n-1}\}^{\mathrm{K}}_{v_p^*,1}$, we have, together with 6), that $v<\beta, n>v_p^*$. Hence v is B_0– α_n–normal.

(B) Suppose now $v_p^*(A_n) = 0$. We define, for every p, a new function $v_p^{\#}$ from FOR into {0,1} in the following way: for every formula B,

1) if A_n is not a subformula of B, then $v_p^{\#}(B) = v_p^*(B)$;

2) if A_n is a subformula of B, then

a) for $B = A_n$, $v_p^{\#}(B) = 1$;
b) for $B = \neg C$, $v_p^{\#}(B) = 1$ iff $v_p^{\#}(C) = 0$;

c) for $B = C \rightarrow D$, $v_p^{\#}(B) = 1$ iff $v_p^{\#}(C) = 0$ or $v_p^{\#}(D) = 1$;

d) for $B = \mathrm{B}C$ or $B = \mathrm{K}C$, $v_p^{\#}(\mathrm{B}) = v_p^*(B)$.

It is now obvious that $v_p^{\#}$ is a semi-valuation. Besides, for $1 \leq i < n$, $v_p^{\#}(A_i) = v_p^*(A_i)$. Since v_p^* is an α_{n-1}–valuation, $v_p^{\#}$ is an α_{n-1}–valuation. We prove that $v_p^{\#}$ is an α_n–valuation. First, since we have $v<\beta, n>v_p^*$, and since v_p^* and $v_p^{\#}$ agree for $i < n$, it follows that $v<\beta, n>v_{p\#}$—from what we get that $\{\alpha_{n-1}\}^{\mathrm{B}}{}_{v,1} = \{\alpha_{n-1}\}^{\mathrm{B}}{}_{vp\#,1}$ and $\{\alpha_{n-1}\}^{\mathrm{K}}{}_{v,1} = \{\alpha_{n-1}\}^{\mathrm{K}}{}_{vp\#,1}$. Let us suppose now that there is $r, s, s < r \leq n$, such that $A_r = \mathrm{B}A_s$ and $v_p^{\#}(A_r) = 0$. Then $v(A_r) = 0$. Since v is an α_n–valuation, from D17 it follows that there is an α_{n-1}–valuation v_r such that $v_r(A_s) = 0$, $v_r(A_m) = 1$ and $v<\beta, n\text{-}1>v_r$. Since the set of true belief and knowledge formulas of v and $v_p^{\#}$ is the same, it follows that $v_p^{\#}<\beta, n\text{-}1>v_r$. If now there is no $r, s, s < r \leq n$, such that $A_r = \mathrm{B}A_s$ and $v_p^{\#}(A_r) = 0$, we get, since v is an α_n–valuation, that there is an α_{n-1}–valuation v_n such that $v_n(A_m) = 1$ and $v<\beta, n\text{-}1>v_n$. Thus $v_p^{\#}<\beta, n\text{-}1>v_n$. It follows that $v_p^{\#}$ is an α_n–valuation.

Now, since $v_p^{\#}(A_n) = 1$, $\{\alpha_n\}^{\mathrm{B}}{}_{v,1} = \{\alpha_n\}^{\mathrm{B}}{}_{vp\#,1}$; and $\{\alpha_n\}^{\mathrm{K}}{}_{v,1} = \{\alpha_n\}^{\mathrm{K}}{}_{vp\#,1}$, and, since $v_p^*(A_m) = v_{p\#}(A_m) = 1$, $v_p^{\#} \models \varepsilon(\{\alpha_n\}^{\mathrm{B}}{}_{v,1})$. Thus $v<\beta, n>v_p^{\#}$ and it follows that for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_n–valuation $v_p^{\#}$ such that $v_p^{\#}(A_q) = 0$ and $v<\beta, n>v_p^{\#}$. That is, v is B_0–α_n–normal.

We prove now that v is B_1– and K_1–α_n–normal. That v is K_1–α_n–normal follows trivially from the fact that it is K_1–α_{n-1}–normal, because $A_n \neq \mathrm{K}A_m$. By induction hypothesis, v is B_1–α_{n-1}–normal, and, from definition 1, we have that for $p = n$, $q = m$, there is an α_{n-1}–valuation v_p such that $v_p(A_m) = 1$ and $v<\beta, n\text{-}1>v_p$. We take the canonical extension v_p^* from v_p to α_n. It is of course an α_n–valuation, and, since $v_p^*(A_m) = 1$, it follows from 3) and 4) that $v<\beta, n>v_p^*$. So v is B_1–α_n–normal.

(3) Let us suppose that, for some $m < n$, $A_n = \mathrm{K}A_m$.

We prove as in **KT5** (with 'K' for '□') that v is K_0– and K_1–α_n–normal. What we should show is that v is B_0– and B_1–α_n–normal as well (remember, true knowledge formulas are also involved, and here we can have one more).

We have again two cases, but, if $v(A_n) = 0$, the proof goes as usual. So let us consider the case where $v(A_n) = 1$. We then have:

1) $\{\alpha_n\}^{\mathrm{K}}{}_{v,1} = \{\alpha_{n-1}\}^{\mathrm{K}}{}_{v,1} \cup \{A_n\}$;

2) $\{\alpha_n\}^{\mathrm{B}}{}_{v+,1} = \{\alpha_{n-1}\}^{\mathrm{B}}{}_{v+,1}$, for every α_{n-1}–valuation v^+;

3) $\varepsilon(\{\alpha_n\}^{\mathrm{B}}{}_{v+,1}) = \varepsilon(\{\alpha_{n-1}\}^{\mathrm{B}}{}_{v+,1})$, for every α_{n-1}–valuation v^+.

Since $v(A_n) = 1$, we have from D17 that:

4) for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_{n-1}–valuation v_p such that $v_p(A_q) = 0$, $v_p(A_m) = 1$ and $v<\beta, n\text{-}1>v_p$.

For each p, let v_p^* be the canonical extension of v_p to α_n. Obviously $v_p^*(A_q) = v_p(A_q)$, and, from (†), v_p^* is an α_n–valuation. It follows that:

5) for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_n–valuation v_p^* such that $v_p^*(A_q) = 0$, $v_p^*(A_m) = 1$ and $v<\beta, n\text{-}1>v_p^*$.

We only need to prove now that $v<\beta, n>v_p^*$; the B_0–α_n–normality follows.

First, since $v_p^*(A_m) = 1$, $v_p^* \models_1 \varepsilon(\{\alpha_{n-1}\}^{\mathrm{B}}{}_{v,1}) \cup \{A_m\}$; thus, from 3),

6) $v_p{}^* \models_1 \varepsilon(\{\alpha_n\}^B{}_{v,1})$.

We have now two cases to consider:

(A) If, now, $v_p{}^*(A_n) = 1$, $\{\alpha_n\}^K{}_{vp^*,1} = \{\alpha_{n-1}\}^K{}_{vp^*,1} \cup \{A_n\}$; $\{\alpha_n\}^K{}_{v,1} = \{\alpha_n\}^K{}_{vp^*,1}$ and since (from 2) $\{\alpha_n\}^B{}_{vp^*,1} = \{\alpha_{n-1}\}^B{}_{vp^*,1}$, and since (from 3) $\varepsilon(\{\alpha_n\}^B{}_{vp^*,1}) = \varepsilon(\{\alpha_{n-1}\}^B{}_{vp^*,1})$, we have, together with 6), that $v<\beta, n>v_p{}^*$. Hence v is B_0– α_n–normal.

(B) Suppose now $v_p{}^*(A_n) = 0$. We define, for every p, a new function $v_p{}^\#$ from FOR into $\{0,1\}$ in the following way: for every formula B,

1) if A_n is not a subformula of B, then $v_p{}^\#(B) = v_p{}^*(B)$;

2) if A_n is a subformula of B, then

a) for $B = A_n$, $v_p{}^\#(B) = 1$;
b) for $B = \neg C$, $v_p{}^\#(B) = 1$ iff $v_p{}^\#(C) = 0$;
c) for $B = C \rightarrow D$, $v_p{}^\#(B) = 1$ iff $v_p{}^\#(C) = 0$ or $v_p{}^\#(D) = 1$;
d) for $B = \mathrm{B}C$ or $B = \mathrm{K}C$, $v_p{}^\#(\mathrm{B}) = v_p{}^*(B)$.

It is now obvious that $v_p{}^\#$ is a semi-valuation. Besides, for $1 \leq i < n$, $v_p{}^\#(A_i) = v_p{}^*(A_i)$. Since $v_p{}^*$ is an α_{n-1}–valuation, $v_p{}^\#$ is an α_{n-1}–valuation. We prove that $v_p{}^\#$ is an α_n–valuation. First, since we have $v<\beta, n>v_p{}^*$, and since $v_p{}^*$ and $v_p{}^\#$ agree for $i < n$, it follows that $v<\beta, n>v_{p\#}$—from what we get that $\{\alpha_{n-1}\}^B{}_{v,1} = \{\alpha_{n-1}\}^B{}_{vp\#,1}$ and $\{\alpha_{n-1}\}^K{}_{v,1} = \{\alpha_{n-1}\}^K{}_{vp\#,1}$. Let us suppose now that there is $r, s, s < r \leq n$, such that $A_r = \mathrm{B}A_s$ and $v_p{}^\#(A_r) = 0$. Then $v(A_r) = 0$. Since v is an α_n–valuation, from D17 it follows that there is an α_{n-1}–valuation v_r such that $v_r(A_s) = 0$, $v_r(A_m) = 1$ and $v<\beta, n\text{-}1>v_r$. Since the set of true belief and knowledge formulas of v and $v_p{}^\#$ is the same (for $i < n$), it follows that $v_p{}^\#<\beta, n\text{-}1>v_r$. Now suppose there is a $r, s, s < r \leq n$, such that $A_r = \mathrm{K}A_s$ and $v_p{}^\#(A_r) = 0$. Then $v(A_r) = 0$, and from D17 we get that there is an α_{n-1}–valuation v_r such that $v_r(A_s) = 0$, $v_r(A_m) = 1$ and $v<\kappa, n\text{-}1>v_r$. Since the set of true knowledge formulas of v and $v_p{}^\#$ is the same (for $i < n$), it follows that $v_p{}^\#<\kappa, n\text{-}1>v_r$. And finally, we have that $v_p{}^*(A_m) = 1$, so $v_{p\#}(A_m) = 1$. It follows that $v_p{}^\#$ is an α_n–valuation.

Now, since $v_p{}^\#(A_n) = 1$, $\{\alpha_n\}^B{}_{v,1} = \{\alpha_n\}^B{}_{vp\#,1}$; and $\{\alpha_n\}^K{}_{v,1} = \{\alpha_n\}^K{}_{vp\#,1}$, and, since $v_p{}^*(A_m) = v_{p\#}(A_m) = 1$, $v_p{}^\# \models \varepsilon(\{\alpha_n\}^B{}_{v,1})$. Thus $v<\beta, n>v_p{}^\#$ and it follows that for every p, every $q, q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$ there is an α_n–valuation $v_p{}^\#$ such that $v_p{}^\#(A_q) = 0$ and $v<\beta, n>v_p{}^\#$. That is, v is B_0–α_n–normal. In a similar way we can prove that v is B_1– α_n–normal, and this completes the proof of the lemma. ∎

Now we have, as consequences of this and the other lemmas the following two propositions:

Corollary. *Let α_n be a normal sequence, v an α_{n-1}–valuation and v_c the canonical extension of v to α_n. Then v_c is an α_n–valuation and, for $1 \leq i \leq n\text{-}1$, $v_c(A_i) = v(A_i)$.*

Theorem T31. *v is an α_n–valuation iff: 1) α_n is a normal sequence; 2) v is a semi–valuation; 3) v is α_n–normal.*

Proofs are exactly the same as in the normal modal logics case, so there is no need to repeat them here.

8.2 Correctness

Having then proved these properties of (α_n–)valuations for **Z5**, we can now move to considering correctness. The notions of satisfiability, validity and semantical consequence are defined in the standard modal logical way (cf. chapter 4). Actually the only big difference now is to prove that the axioms of **Z5** are valid under this semantics; the rest follows in the same old way. So let us get to the following

Lemma L27. *Let v be an α_n–valuation; then, for $1 \leq i \leq n$, if A_i is an axiom of Z5 then $v(A_i) = 1$.*

Proof. If A_i is an axiom, then it is either an axiom from **PL**, and it follows from the fact that v is a semi–valuation that $v(A_i) = 1$, or it is one of the modal axiom schemas. Now, if A_i is one of the pure belief axioms, the proof is the same as in **KD45** (with 'B' instead of '□'), and, if A_i is one of the pure knowledge axioms, the proof goes as in **KT5** (with 'K' instead of '□'). We consider then only the cases of the mixed axioms. In **Z5** there is just one of them, m, so suppose $A_i = \mathrm{K}A \rightarrow \mathrm{B}A$. Suppose $v(A_i) = 0$. Then we have $v(\mathrm{K}A) = 1$ and $v(\mathrm{B}A) = 0$. From the normality lemma it follows that for every p, every q, $q < p \leq n$, such that $A_p = \mathrm{B}A_q$ and $v(A_p) = 0$, there is an α_n–valuation v_p such that $v_p(A_q) = 0$ and $v{<}\beta, n{>}v_p$. Thus $v_p(A) = 0$. Now $v{<}\beta, n{>}v_p$ means, among other things, that $\{\alpha_n\}^{\mathrm{K}}{}_{v,1} = \{\alpha_n\}^{\mathrm{K}}{}_{v_p,1}$. So $v_p(\mathrm{K}A) = 1$. Since, now, v_p is K_1–α_n–normal, we have $v_p(A) = 1$, a contradiction. So, for every valuation v, $v(A_i) = 1$. ■

Now we have no trouble to show that, if A is an axiom of **Z5** and v is a valuation, that $v(A) = 1$. For the next lemma and its corollary, too, the proof is the same as in the normal logics case (with the care of substituting 'B' for '□'). The same holds, finally, for the Correctness Theorem, so we can jump without delay to the next section.

Lemma L28. *For all n, all i, $1 \leq i \leq n$, if v is an α_n–valuation and $\vdash_L A_i$, then $v(A_i) = 1$.*

Corollary. *If $\vdash A$ then $\models A$.*

Theorem T32. (*Correctness Theorem*) *If $\Gamma \vdash A$ then $\Gamma \models A$.*

8.3 Completeness

Completeness is again easy to prove making use of saturated sets—which we already have defined for knowledge and belief on part I. The main task here is to proof that characteristic functions of saturated sets are α_n–valuations; the rest follows smoothly in the good old way. So let us consider our

Theorem T33. *For every A–saturated set Δ and every normal sequence α_n, the characteristic function f of Δ is an α_n–valuation.*

Proof. First of all, it is easy to prove by P5 that

(†) The characteristic function f of Δ is a semi–valuation.

We now prove the theorem by induction on n. If $n = 1$, the property follows from (†) above. Let us suppose $n > 1$.

(1) If, for every $m < n$, $A_n \neq \mathrm{B}A_m$, $A_n \neq \mathrm{K}A_m$, f is trivially an α_n–valuation.

(2) For some $m < n$, $A_n = \mathrm{B}A_m$.

(I) $f(A_n) = 0$. Then $A_n \notin \Delta$, $\Delta \nvdash \mathrm{B}A_m$. From P7, there is an A_m-saturated set Θ such that $\varepsilon(\Delta^{\mathrm{B}}) \subseteq \Theta$. Let f_Θ be the characteristic function of Θ. By the induction hypothesis, f and f_Θ are α_{n-1}–valuations. We also have, since Θ is A_m-saturated, that $f_\Theta(A_m) = 0$. Now, $\{\alpha_{n-1}\}^{\mathrm{B}}{}_{f,1} \subseteq \Delta$, thus $\varepsilon(\{\alpha_{n-1}\}^{\mathrm{B}}{}_{f,1}) \subseteq \varepsilon(\Delta^{\mathrm{B}}) \subseteq \Theta$; thus $f_\Theta \vDash \varepsilon(\{\alpha_{n-1}\}^{\mathrm{B}}{}_{f,1})$. Now, from P8, $\Delta^{\mathrm{B}} \subset \varepsilon(\Delta^{\mathrm{B}})$. It is then easy to see that $\{\alpha_{n-1}\}^{\mathrm{B}}{}_{f,1} = \{\alpha_{n-1}\}^{\mathrm{B}}{}_{f_\Theta,1}$. Now, from P8, $\Delta^{\mathrm{K}} \subseteq \varepsilon(\Delta^{\mathrm{B}})$ (because $\vdash \mathrm{K}A \rightarrow \mathrm{BK}A$). It is then easy to see that $\{\alpha_{n-1}\}^{\mathrm{K}}{}_{f,1} = \{\alpha_{n-1}\}^{\mathrm{K}}{}_{f_\Theta,1}$. We thus can say that $f{<}n\text{-}1{>}f_\Theta$; hence f is an α_n–valuation.

(II) $f(A_n) = 1$. So $\mathrm{B}A_m \in \Delta$, $\Delta \vdash \mathrm{B}A_m$. Let us suppose there is some p, some q, $q < p \leq n$ such that $A_p = \mathrm{B}A_q$ and $f(A_p) = 0$. From P7, there is an A_q–saturated set Θ such that $\varepsilon(\Delta^{\mathrm{B}}) \subset \Theta$. Let f_Θ be the characteristic function of Θ. By the induction hypothesis, f and f_Θ are α_{n-1}–valuations. With an analogous argument as in case I), we show that $f{<}n\text{-}1{>}f_\Theta$. Since Θ is A_q–saturated, $f_\Theta(A_q) = 0$ and, since $A_m \in \varepsilon(\Delta^{\mathrm{B}})$, $f_\Theta(A_m) = 1$.

Now it follows from lemma 5, since $\Delta \vdash \mathrm{B}A_m$, that there is an $\neg A_m$–saturated set Θ such that $\varepsilon(\Delta^{\mathrm{B}}) \subseteq \Theta$. We prove in a similar way that f_Θ is an α_{n-1}–valuation, $f{<}\text{n-1}{>}f_\Theta$ and $f_\Theta(A_m) = 1$.

(3) For some $m < n$, $A_n = \mathrm{K}A_m$. Proof similar to (2) and to the KT5 case. ■

Finally, the proof of this lemma's corollary, and of the Completeness Theorem, suffer no change from the modal logic case:

Corollary. *v is a valuation iff v is the characteristic function of some saturated set Δ.*

Theorem T34. (*Completeness Theorem*) *If $\Gamma \vDash A$ then $\Gamma \vdash A$.*

8.4 GTTs for Z5

The last thing needing consideration in this chapter, before we go on, is the definition, based on the semantics just seen, of GTTs for **Z5**. There are of course differences in comparison to what we have done to **K**, but they show up only in the case of the modal operators, that is, cases (d) and (e) of the old definition will have to be changed. I'll give only the important part of the definition:

Definition D19. Let α_n be a normal sequence. A *generalized truth-table* (GTT) for α_n is a function $T[\alpha_n] : \{\alpha_n\} \times J(\alpha_n) \rightarrow \{0,1\}$, where:

1) for $n = 1$, $J(\alpha_1) = \{1,2\}$, $T[\alpha_1](A_1, 1) = 1$ and $T[\alpha_1](A_1, 2) = 0$;

2) for $n > 1$, and $J(\alpha_{n-1}) = \{1,\dots,q\}$:

(a) *propositional variables:* as in **K**;

(b) *negation:* as in **K**;

(c) *implication:* as in **K**;

(d) if $A_n = \mathrm{B}A_k$, $k < n$, then for every $j \in J(\alpha_{n-1})$:

I) for $u \in \{1,0\}$, let $\alpha(u, j, n\text{-}1) = \{j' \in J(\alpha_{n-1})$: $T[\alpha_{n-1}](A_k, j') = u$ and, for every r, $1 \leq r \leq n$, if $A_r = \mathrm{B}A_s$ and $T[\alpha_{n-1}](A_r, j) = 1$, then $T[\alpha_{n-1}](A_s, j') = 1$; if $A_r = \mathrm{B}A_s$ or $A_r = \mathrm{K}A_s$,$T[\alpha_{n-1}](A_r, j) = T[\alpha_{n-1}](A_r, j')\}$;

II) for every p, every q, $q < p < n$ such that $A_p = \mathrm{B}A_q$ and $T[\alpha_{n-1}](A_p, j) = 0$, let $\beta(p, j, n\text{-}1) = \{j' \in J(\alpha_{n-1})$: $T[\alpha_{n-1}](A_q, j') = 0$, $T[\alpha_{n-1}](A_k, j') = 1$ and, for every r, $1 \leq r \leq n$, if $A_r = \mathrm{B}A_s$ and $T[\alpha_{n-1}](A_r, j) = 1$, then $T[\alpha_{n-1}](A_s, j') = 1$; if $A_r = \mathrm{B}A_s$ or $A_r = \mathrm{K}A_s$,$T[\alpha_{n-1}](A_r, j) = T[\alpha_{n-1}](A_r, j')\}$;

III) let $\{j_1,\dots j_m\} \subseteq J(\alpha_{n-1})$ such that:

1) $j_{m'} < j_{m''}$ if $m' < m''$;

2) $j_{m'} \in \{j_1,\dots j_m\}$ if $\alpha(j_{m'}, n\text{-}1) \neq \emptyset$ and, for every $p < n$, $\beta(p, j_{m'}, n\text{-}1) \neq \emptyset$.

Then $J(\alpha_n) = \{1,\dots,q,\dots,q+m\}$ and:

i) for $i < n, j \leq q$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i < n, j = q + m'$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j_{m'})$;

iii) for $i = n$, j such that $\alpha(0, j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 1$;

iv) for $i = n$, j such that $\alpha(1, j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 0$;

v) for $i = n$, j such that $\alpha(0, j, n\text{-}1) \neq \emptyset$ and, for some $p < n$, $\beta(p, j, n\text{-}1) = \emptyset$, or $\alpha(1, j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) =0$;

vi) for $i = n$, j such that $\alpha(0, j, n\text{-}1) \neq \emptyset$, $\alpha(1, j, n\text{-}1) \neq \emptyset$ and, for every $p < n$, $\beta(p, j, n\text{-}1) \neq \emptyset$, in which case, for some $m' \in \{1,\dots,m\}$, $j = j_{m'}$ or $j = q + m'$,

1) if $j = j_{m'}$ then $T[\alpha_n](A_i, j) =1$;

2) if $j = q + m'$ then $T[\alpha_n](A_i, j) =0$.

(e) if $A_n = \mathrm{K}A_k$, $k < n$, then for every $j \in J(\alpha_{n-1})$:

I) let $\gamma(j, n\text{-}1) = \{j' \in J(\alpha_{n-1})$: $T[\alpha_{n-1}](A_k, j') = 1$ and, for every r, $1 \leq r \leq n$, if $A_r = \mathrm{K}A_s$,$T[\alpha_{n-1}](A_r, j) = T[\alpha_{n-1}](A_r, j')\}$;

II) for every p, every q, $q < p < n$ such that $A_p = \mathrm{B}A_q$ and $T[\alpha_{n-1}](A_p, j) = 0$, let $\delta(p, j, n\text{-}1) = \{j' \in J(\alpha_{n-1})$: $T[\alpha_{n-1}](A_q, j') = 0$, $T[\alpha_{n-1}](A_k, j') = 1$ and, for every r, $1 \leq r \leq n$, if $A_r = \mathrm{B}A_s$ and $T[\alpha_{n-1}](A_r, j) = 1$, then $T[\alpha_{n-1}](A_s, j') = 1$; if $A_r = \mathrm{B}A_s$ or $A_r = \mathrm{K}A_s$,$T[\alpha_{n-1}](A_r, j) = T[\alpha_{n-1}](A_r, j')\}$;

III) let $\{j_1,\dots j_m\} \subseteq J(\alpha_{n-1})$ such that:

1) $j_{m'} < j_{m''}$ if $m' < m''$;

2) $j_{m'} \in \{j_1,\dots j_m\}$ if $\alpha(j_{m'}, n\text{-}1) \neq \emptyset$, for every $p < n$, $\beta(p, j_{m'}, n\text{-}1) \neq \emptyset$, and $T[\alpha_{n-1}](A_k, j) = 1$.

Then $J(\alpha_n) = \{1,\dots,q,\dots,q+m\}$ and:

i) for $i < n, j \leq q$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j)$;

ii) for $i < n, j = q + m'$, $T[\alpha_n](A_i, j) = T[\alpha_{n-1}](A_i, j_{m'})$;

iii) for $i = n$, j such that $\gamma(j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 1$;

iv) for $i = n, j$ such that $\alpha(j, n\text{-}1) \neq \emptyset$ and, for some $p < n$, $\delta(p, j, n\text{-}1) = \emptyset$, $T[\alpha_n](A_i, j) = 0$;

v) for $i = n$, and $T[\alpha_{n\text{-}1}](A_k, j) = 0$, $T[\alpha_n](A_i, j) = 0$;

vi) for $i = n, j$ such that $\alpha(j, n\text{-}1) \neq \emptyset$, for every $p < n$, $\delta(p, j, n\text{-}1) \neq \emptyset$, and $T[\alpha_{n\text{-}1}](A_k, j) = 1$, in which case, for some $m' \in \{1,...,m\}$, $j = j_{m'}$ or $j = q + m'$,

1) if $j = j_{m'}$ then $T[\alpha_n](A_i, j) = 1$;

2) if $j = q + m'$ then $T[\alpha_n](A_i, j) = 0$.

Now there is probably not very much to explain: this definition just mirrors what we've already done in the valuation definitions for **Z5** in the preceding sections. We only have to prove that things work, and the only result we have to get is the following

Lemma L29. *Let α_n be a normal sequence, and v a valuation. Then there is $j \in J(\alpha_n)$ such that, for $1 \leq i \leq n$, $v(A_i) = T[\alpha_n](A_i, j)$.*

Proof (cf. the normal modal logics case.) ■

Lemma L30. *For every normal sequence α_n, if, for some $i \leq n$, $\vdash_K A_i$ then for every $j \in J(\alpha_n)$, $T[\alpha_n](A, j) = 1$.*

Proof. (cf. the normal modal logics case.) ■

Theorem T35. *$\vdash A$ iff for every normal sequence α_n, where $A = A_i$, $1 \leq i \leq n$, and for every $j \in J(\alpha_n)$, $T[\alpha_n](A_i, j) = 1$.*

Proof. (cf. the normal modal logics case.) ■

As we then see, **Z5** is also decidable by GTTs.

Intermezzo 2

So Part II is finished, and now, with our thus acquired knowledge about valuation semantics and generalized truth tables, we can move to the third Part of this dissertation, where we are going to try putting into practice a little bit of what we learned. The next three chapters will thus discuss some implementations. Chapter 9 presents a GTT-builder for an example EDL (Z5). Building a whole table, however, is something costly in time and memory, so in Chapter 10 we take another EDL (ZP5) and present for it a tableau-like theorem prover. The last Chapter, 11, shows an implementation of the algorithm for characterizing minimal belief states from Part I.

III

Implementations

9

Implementing a GTT Builder for Z5

Oh, I am a C programmer and I'm okay
I muck with indices and structs all day
And when it works, I shout hoo-ray
Oh, I am a C programmer and I'm okay.

In this chapter we are going to examine a simple C program which implements the construction of GTTs for Z5 formulas. This is going to be a very straightforward implementation of the GTT definition which we have just given in chapter 8. It is not intended to be a real, fast theorem prover for Z5, even if its performance it's not *that* bad—it is more a 1 to 1 implementation of the semantics, mainly pretending to show how things can be done. It has been chosen to be close to the definition, not to be fast. (Surprisingly, however, it can be very fast in certain cases, when compared to other implementations.)

The program, which is called `GTT.Z5`, has three main parts, which are split over several files. The main loop of the program does the following things:

- reads a formula (string) from the standard input;
- parses the formula, that is, the read string is transformed into a tree-like internal representation;
- calls a function which constructs the table;
- and, finally, printes the output on the screen.

I am not going to discuss every little thing on the program (for example, parsing and printing routines are just going to be mentioned), but rather the building of the table. So let us begin.

9.1 Data Structures and `main()`

We examine first the macros and globals, which are to be found in the file "`macros.h`".

9.1.1 Macros

```
#define STR_LEN         256
```

This is the maximum allowed length of the input strings.

```
#define MAX_LINES        64
#define MAX_ROWS         30
```

Ditto for the maximum number of lines and rows in the GTT. We are going to have the tables statically defined as a bidimensional array, where the rows will held places for the wffs. Another alternative, of course, would be using dynamic memory allocation, but I think that with a statically defined array things are easier to grasp.

The next lines define, first, a "pointer" to NULL: since arrays in C begin with 0, and since, as we'll see, contents of fields will be pointing to other places of the arrays, we have to know when we are meaning, say, location 0, or meaning nothing. In the other lines connectives are given an internal representation code by means of integers. For atomic propositions ('a' till 't') we'll use its ASCII-code.

```
#define CNULL            (-1)        /* a "pointer" to NIL   */
#define UND              5           /* conjunction */
#define ODR              6           /* disjunction */
#define IMP              7           /* implication */
#define NEG              8           /* negation */
#define EQU              9           /* biconditional */
#define KNW              10          /* knowledge */
#define BEL              11          /* belief */
```

In the next lines we define two kinds of "atomic propositions": on input, when the wff is being read and parsed, atoms are small letters. But internally, after that, they'll be everything *but* connectives—we have to make just one check, instead of two. Thus the program will run a little bit faster.

```
#define P_ATOMIC( c )   ( c >= 'a' && c <= 't' )       /* atoms on input */
#define ATOMIC( c )     ( c > BEL )                    /* atoms internally */
```

The last macro line is just a definition for the parsing mechanism. I'll explain that later.

```
#define INIT_NODE(a,b,c)        WFF[wf_ptr][0] = a;     \
                                WFF[wf_ptr][1] = b;     \
                                WFF[wf_ptr][2] = c
```

9.1.2 Globals

We'll also be using some global variables, which we get on the next lines:

```
short WFF[STR_LEN][3],
      wf_ptr,
      TBL[MAX_ROWS][MAX_LINES];

char  theWff[STR_LEN];

long  k1, k2, k3, k4;
```

- WFF[STR_LEN][3] is our bidimensional arrray where the internal tree-like representation of formulas is stored; wf_ptr points to the next free place on it. The parsing mechanism takes as input a string, like a&b->¬av¬b

and converts it into a tree, which is stored in WFF like

0	1	2	3	4	5	6
a	b	&	¬	¬	v	->
		0	0	1	3	2
		1			4	5

fig. 24

The rightmost used place is, of course, the "top" node of the tree. Numbers under each operator denote the row in the array where the arguments are to be found. So, for example, the disjunction in 5 has as the left disjunct the wff of row 3, and as the right disjunct, the one in row 4.[48]

The table (TBL) is also a bidimensional array. I have chosen a small number of rows and lines, but this can be easily changed to suit one's needs.

Finally, the `long` integers are used for performance measurements.

Now the `main()` function has the following code:

```
main()
{
      short i, j, k, u;

      printf("\n*****************************************************");
      printf("\n*                GTT BUILDER FOR Z5                 *");
      printf("\n*                 Cesar A. Mortari                  *");
      printf("\n*                  V1.0, May 1990                   *");
      printf("\n*****************************************************");
      printf("\n\nSyntax:\n a..t (variables), ¬, K, B, &, v, ->, <->\n");
      printf("\nTo exit type ';'<CR>\n");
      printf("\nPlease type in a formula:\n");
```

This was just printing startup information. Now we enter the main program loop (with `for (;;)`).First we have two initializations: `wf_ptr` is set to `0` (next free row in WFF), and `WFF[0][0]` is set to CNULL (to ensure that the parser won't think that this place, automatically filled with a `0` on startup[49], is pointing to row zero...)

```
            wf_ptr = 0;
            WFF[0][0] = CNULL;
```

The next step is to read the formula of which we want to have a table. If the input is ';', that means "end the program",[50] else a small routine remove all blanks from `theWff`:

```
            gets( theWff );
            if ( theWff[0] == ';')
                  return 1 ;
            i = 0;
```

[48] In reality we don't have, for instance in row 6, the characters '-' and '>' stored there, but rather the value 6, which, by macro definition, represents the arrow. Similarly for the other symbols.

[49] I'm implementing this with the THINK C Compiler, version 4.0, which has this characteristic.

[50] With the THINK C Compiler this is actually unnecessary, because a standard console window is provided, together with a "Quit" menu option. With other compilers which don't provide this option, one has to introduce a way of interrupting the main loop.

```
j = 0;
while ( theWff[i] != '\0' )
{
      if ( theWff[i] != ' ' )
           theWff[j++] = theWff[i];
      ++i;
}
theWff[j] = '\0';
```

Next we initialize the time counting, and call the parser (with `formula()`) to convert the string into a tree, at the same time checking if the syntax was correct. If everything is OK, we get the time used in parsing and go build the table (`make_table()`). After that the contents of the `WFF` array are displayed (`displayWFF()`) and the table is printed. The last lines print then the time used for the different routines on the screen.

```
k1 = TickCount();         /* initialize time counting */

if ( formula( 0, &u, &i ) && theWff[u] == '\0' )
{
      k2 = TickCount();
      k = make_table();
      k3 = TickCount();
      displayWFF();
      print_table( wf_ptr, k );
      k4 = (( TickCount() - k3 ) * 100) / 6;
      k3 = ((k3-k2)*100)/6;
      k2 = ((k2-k1)*100)/6;
      printf("Parsing time: %ld ms", k2);
      printf("\tMake table: %ld ms",k3);
      printf("\nPrint table: %ld ms",k4);
      printf("\tTotal time: %ld ms\n\n",k2+k3+k4);
}
```

If we had a syntax error, of course, nothing applies, so we print an error message and start again.

```
else
      printf("\nSyntax error...\n\n");
```

We are now ready to consider more details of the program. The other routines are in different files, which are included (with `#include <file>`) just before `main()`. They are

```
#include "macros.h"
#include "prototypes.h"
#include "parser.h"
#include "outputs.h"
#include "table.h"
```

`prototypes.h` is just a small file containing the prototypes of all GTT.Z5 functions—similar to the "forward" declarations in Pascal procedures and functions. In C it is normally not needed, but see listing in Appendix B.

9.3 parser.h

This is the file containing the parsing functions. I won't discuss it here in detail, just give general information. It is an adaptation from a parser once written in Prolog by Franz Guenthner. I've changed it here to use my data structures, as well as introducing a mechanism to check if a node is already there. For instance, if we had a formula like a&b -> a&b, we could end up having a tree with twice the same node, as following:

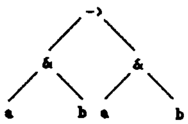

fig. 25

In order not to get repeated wffs when building the tables, which is totally unnecessary, we need a checking routine that looks, before creating a node, to see if it is maybe already there. We would then have the following representation:

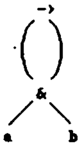

fig. 26

And that's all about parsing. See listing in Appendix B, if you are interested. It needs some improving, too.

9.3 outputs.h

Little to say about this: just two functions which, first, display the contens of WFF and, second, print the table (that is, the TBL array). See listing in Appendix B.

9.4 table.h

And finally, the routines to build the table. We first have the code of the make_table() function, which we are going to discuss now:

```
short make_table() /*
-------------------*/
{
      int   i, row, line, ad_lines;
```

i is a loop variable; row and line denote, respectively, the current row and line which we are computing. ad_lines is the number of added lines in the case of knowledge and belief formulas (correspond to the *m* parameter in the definition of GTT (as in D19.d, for example).

Now the first step is to initalize the first row and first two lines of the table (the first row is *always* a propositional variable...). The current (bottom) line number is then set to 1, and ad_lines is initialized to 0. Just note that the actual number of lines in the table is equal to line + 1, because lines are numbered from 0 to line.

```
TBL[0][0] = 1;
TBL[0][1] = 0;
line = 1;
ad_lines = 0;
```

Now we enter the Main Table Loop: we do a loop creating each new row, starting with 1 (the row 0 already contains a propositional variable) and going until wf_ptr-1 (which points to the top node of our formula). If the number of rows or lines gets greater than MAX_ROWS or MAX_LINES, the loop is stopped and we exit with 0. Else we make a switch on the main (current wff) connective, and act accordingly.

```
for ( row = 1; row < wf_ptr; ++row )
{
      if ( row >= MAX_ROWS || line >= MAX_LINES )
      {
            printf("\n\n***** ERROR : TABLE TOO LARGE! ******");
            return 0;
      }
      switch( WFF[row][0] )
      {
```

The cases in which the operators are boolean ones, we just do a boolean computing of its argument(s), starting with line 0 and going until line.

```
      case NEG :
            for ( i = 0; i <= line; ++i )
                  TBL[row][i] = !TBL[WFF[row][1]][i];
            break;
      case UND :
            for ( i = 0; i <= line; ++i )
                  TBL[row][i] = TBL[WFF[row][1]][i]
                  && TBL[WFF[row][2]][i];
            break;
      case ODR :
            for ( i = 0; i <= line; ++i )
                  TBL[row][i] = TBL[WFF[row][1]][i]
                  || TBL[WFF[row][2]][i];
            break;
      case IMP :
            for ( i = 0; i <= line; ++i )
                  TBL[row][i] = !TBL[WFF[row][1]][i]
                  || TBL[WFF[row][2]][i];
            break;
      case EQU :
            for ( i = 0; i <= line; ++i )
                  TBL[row][i] = (TBL[WFF[row][1]][i]
                                == TBL[WFF[row][2]][i]);
            break;
```

Now we have to consider the epistemic operators, where things are slightly more complicated. Let us begin with knowledge (and let's begin on the left margin, for clarity):

```
case KNW :
{
      for ( i = 0; i <= line; ++i )
            if ( TBL[WFF[row][1]][i] == 0 )
                  TBL[row][i] = 0;
            else
            {
                  if ( gammaNE( row, line, WFF[row][1], 0, i ) )
                  {
                        if ( deltasNE( row, line, WFF[row][1], 0, i ) )
                        {
                              split_lines( row, line+1+ad_lines, i );
                              ++ad_lines;
                        }
                        else
                              TBL[row][i] = 0;
                  }
                  else
                        TBL[row][i] = 1;
            }
      line = line + ad_lines;
      ad_lines = 0;
      break;
}
```

Let us examine ali that. Let us suppose we have some *KA* as our wff in the current row. First, of course, we do a `for` loop to compute the value of *KA* in each line `i`. And this goes as follows: first, if `TBL[WFF[row][1]][i] == 0`—that is, if *A* has a value `0` in line `i`, obviously *KA* will have to get `0`, because we are in **S5** for knowledge, and reflexivity holds. This corresponds to clause (e.iii) in the GTT definition D18. Else, if *A* has a value 1, there is more that we have to check. The function `gammaNE()` checks if γ(i, row) is not empty (cf. D18.e.iii-v). That is, wheter there is a line `j` giving 0 to A and being "accessible" to the current line `i`. If there is not, then *KA* gets the value 1 (cf. D18.e.iii). Else we check the normality conditions (for every p, every *q*, $q < p < n$, which is a modal wff and gets zero, etc). This is what the function `deltasNE()` (δ is non empty) does. If it fails, then KA gets 0 (cf. D18.e.iv). Else everything checks, and we then arrive to the case where we have to split the line, what we do with `split_lines()`: the current line is copied at the bottom of the table, and *KA* gets then 1 in the old line, and zero in the new one. Just like the definitions. We then increment `ad_lines` to signal that a line was added.

Exiting the for loop, we set the global line number to its old value, plus the added lines, and reinitialize `ad_lines` to zero.

I am not going to discuss here the code of subroutines like `gammaNE()`, and so on. But see listing in Appendix B.

The case of belief is now similar. We have of course the belief correspondents of `gammaNE()` and `deltasNE()`, that is, `alphaNE()` and `betasNE()`. The first check does not more cares for reflexivity, because here we have seriality. The rest is pretty much the same.

```
case BEL :
{
      for ( i = 0; i <= line; ++i )
            if ( !alphaNE( row, line, WFF[row][1], 0, i, 1 ) )
                  TBL[row][i] = 0;
            else
            {
                  if ( alphaNE( row, line, WFF[row][1], 0, i, 0 ) )
                  {
                        if ( betasNE( row, line, WFF[row][1], 0, i ) )
                        {
```

```
                    split_lines( row, line+1+ad_lines, i );
                    ++ad_lines;
                }
                else
                    TBL[row][i] = 0;
            }
            else
                TBL[row][i] = 1;
        }
    line = line + ad_lines;
    ad_lines = 0;
    break;
}
```

And, last but not least, we have to consider the case where new propositional variables appear!

```
            default :        /* Atomic propositions */
            {
                copy_lines( row, line + 1 );
                for ( i = 0; i <= line; ++i )
                    TBL[row][i] = 1;
                line = 2 * (line + 1) - 1;
                while ( i <= line )
                    TBL[row][i++] = 0;
            }
        }
```

When we get a variable, we have of course to double the number of lines. This is what the routines do: `copy_lines()` make of course a copy of the old lines; then the values 1 and 0 are set, and the number of lines is updated.

After exiting the main table loop, we return the number of lines, which will be needed by `print_table()`.

```
    }
    return ((short) line+1);
}
```

And that's it.

9.5 A working session with GTT.Z5

Now some examples from what happens when the program runs: let us type some formulas and see what happens. I ran it on my Macintosh Plus, with 1MB RAM:

```
****************************************************
*                 GTT BUILDER FOR Z5               *
*                  Cesar A. Mortari                *
*                  V1.0, May 1990                  *
****************************************************

Syntax:
       a..t (variables), ¬, K, B, &, v, ->, <->

To exit type ';'<CR>

Please type in a formula:
Ka->Ba
```

```
[a][K][B][>]
[ ][0][0][1]
[ ][ ][ ][2]

*** TABLE ***
[0][1][2][3]
|1||1||1||1|
|0||0||1||1|
|1||0||1||1|
|0||0||0||1|
|1||0||0||1|

Parsing time: 0 ms      Make table: 0 ms
Print table: 1550 ms    Total time: 1550 ms

¬Ka->K¬Ka

[a][K][-][K][>]
[ ][0][1][2][2]
[ ][ ][ ][ ][3]

*** TABLE ***
[0][1][2][3][4]
|1||1||0||0||1|
|0||0||1||1||1|
|1||0||1||1||1|

Parsing time: 0 ms      Make table: 16 ms
Print table: 1066 ms    Total time: 1082 ms

Ba->a

[a][B][>]
[ ][0][1]
[ ][ ][0]

*** TABLE ***
[0][1][2]
|1||1||1|
|0||1||0|
|1||0||1|
|0||0||1|

Parsing time: 0 ms      Make table: 16 ms
Print table: 1133 ms    Total time: 1149 ms

Ba->KBa

[a][B][K][>]
[ ][0][1][1]
[ ][ ][ ][2]

*** TABLE ***
[0][1][2][3]
|1||1||1||1|
|0||1||1||1|
|1||0||0||1|
|0||0||0||1|
|1||1||0||0|
|0||1||0||0|

Parsing time: 0 ms      Make table: 0 ms
Print table: 1316 ms    Total time: 1316 ms

Ka->BKa

[a][K][B][>]
[ ][0][1][1]
[ ][ ][ ][2]

*** TABLE ***
[0][1][2][3]
|1||1||1||1|
|0||0||0||1|
|1||0||0||1|

Parsing time: 0 ms      Make table: 0 ms
Print table: 1050 ms    Total time: 1050 ms

Ba->BKa

[a][B][K][B][>]
[ ][0][0][2][1]
[ ][ ][ ][ ][3]

*** TABLE ***
[0][1][2][3][4]
|1||1||1||1||1|
|0||1||0||0||0|
|1||0||1||1||1|
|0||0||0||0||1|

Parsing time: 0 ms      Make table: 16 ms
Print table: 1116 ms    Total time: 1182 ms
```

As we see, the right results are coming out. Some formulas are theorems of **Z5**, and others not, as we can see b. I suggest the reader try him- or herself the program.

10

A tableau-like theorem prover for ZP5

Third Law of Computer Programming:

Any given improvement costs more and takes longer.

In this chapter we'll also be discussing a C program which implements a theorem prover for **ZP5**, but in a more efficient way than what was done in the last chapter for **Z5**. The idea is to adopt a refutation proof procedure, instead of building a whole truth table. Basically we'll take the tableau extension rules for the classical case, and enlarge them adding rules to cope with the knowledge and belief operators. The intuition for these rules comes to the conditions defined in valuation semantics.[51]

This program is called `TTP.ZP5`, and it is very similar, on the overall structure, to the one considered in the previous chapter (`GTT.Z5`). The parsing mechanism is not exactly the same—two important changes were made because the proof procedure makes other assumptions. The big change, ofcourse, is that we no longer build and display a GTT, but use this tableau procedure to get a simple "yes" or "no" as whether some wff is valid in **ZP5** or not. The main loop of the program does the following things:

- reads a formula (string) from the standard input;
- parses the formula, transforming the read string into a tree-like internal representation;
- runs the tableau proof procedure; and
- prints the answer on the screen.

10.1 Data Structures and `main()`

We examine first some macros and the globals.

[51] `TTP.ZP5` is based on an older version of `FTL`, a tableau theorem prover for the classical propositional logic written by J. Hudelmaier and myself, ([HdM89]), but it underwent extensive rewriting to cope with modalities. In particular, I would like to mention that some tricks to cut branches used in later versions of `FTL` are not being implemented here.

10.1.1 Macros

Some things are going to remain the same as in the previous program (like length of strings). We don't need MAX_LINES and MAX_ROWS anymore, but as new stuff have the following:

```
#define EMPTY              (-2)
#define LMK                (-3)         /* left marker */
#define RMK                (-4)         /* right marker */
#define L_SUB              1
#define R_SUB              2
#define UND                6            /* negation */
#define KNW                7            /* knowledge */
#define BEL                8            /* belief */
#define U_KNW              9            /* knowledge, used */
#define U_BEL              10           /* belief, used */
#define USD                11
```

As the reader may have noticed, also the "codes" for the operators has changed: first, from the classical functors, only conjunction remained. The reason is that formulas will be being rewritten on parsing, so we eliminate negations, disjunctions and implications. As a side effect of this policy, I'll be leaving equivalences out for simplicity (cf. later on parsing). And second, for knowledge and belief we also have an U_KNW and a U_BEL: this shall show the search mechanism that the corresponding wffs were already processed in the branch. USD has the same function, but for other wffs.

Now, if we take a look at the listing of macros.h (in Appendix C) we'll find that there is much more. I'll discuss some of them macros when opportunity arises; I can't do this here without first explaining how the program is supposed to work.

10.1.2 Globals

The global variables which we'll be using are:

```
short     WFF[STR_LEN][3],
          PRF[STR_LEN][3],
          ALT[STR_LEN],
          BLF[STR_LEN],
          BCKT[STR_LEN][4],
          MOD[STR_LEN][2],
          wf_ptr,
          prf_ptr,
          mod_ptr,
          alt_ptr,
          bckt_ptr;

char      theWff[STR_LEN];       /* input formula */

long      k1, k2;                /* for time measurement */
```

WFF and wf_ptr are know from the last program. But, as one can see, there are some arrays more, in place of TBL (which stored the table). PRF is where we are going to store the current branch. ALT (with help of BLF) has to do with the possible alternate worlds we still have to consider. BCKT has to do with backtraking: when we get some branching, we store the other possibility (together with the actual state of affairs) on BCKT and, after getting a contradiciton, return to it and try the other continuation. MOD stores the modalized formulas of the branch. The other `short` integers are place-pointers for each of these arrays, and the `long` integers are used for performance measurements.

The code of the `main()` function is very similar to the one in the previous program, so I won't bother to repeat it here whole. Basically, there is some startup information being outputed on the screen, after what one reads a wff (or '`;`' to end the program), and proceeds to parse it. The only change worth mentioning appears in the following piece of code:

```
if ( formula( 0, &u, &i ) && theWff[u] == '\0' )
{
        if ( KB( WFF[i][0] ) )
        {
                ++mod_ptr;
                MOD[0][0] = -WFF[i][0];
                MOD[0][1] = i;
        }
        else
        {
                ++prf_ptr;
                PRF[0][0] = -WFF[i][0];
                PRF[0][1] = i;
                PRF[0][2] = 0;
        }
        if ( tableau())   /* all branches were closed...     */
        {
                k2 = (( TickCount() - k1 ) * 100) / 6;
                putchar( 'y' );
                putchar( 'e' );
                putchar( 's' );
        }
        else        /* some open branch - tableau() returned zero  */
        {
                k2 = (( TickCount() - k1 ) * 100) / 6;
                putchar( 'n' );
                putchar( 'o' );
        }
        printf("\nTime: %ld ms\n\n\n", k2);

}
```

After having succesfully parsed the wff, there is a call to the proof procedure with `tableau()`, but, before this, we have to add the formula to be (dis)proven to the branch. As I explain later on, the program keeps modalized and unmodalized formulas of the branch in different places. That explains the line `if ( KB( WFF[i][0] ) )` in which `i` is the address of the wff. The macro KB only checks whether the wff's main operator is a modal one. If yes, the branch begins in MOD, else in PRF. Now `tableau()` is called, and runs as long as there is something to do, only stopping and returning 1 if all branches are closed, or 0 if there is an open branch which cannot be further processed.

If we had a syntax error, of course, nothing applies, so we print an error message and start again.

We are now ready to consider more details of the program. The other routines are in different files, which are included (with `#include <file>`) just before `main()`. They are

```
#include "macros.h"
#include "prototypes.h"
#include "parser.h"
#include "tableau.h"
```

10.2 parser.h

Here we find the parsing functions. I won't discuss it again in detail, just remark that there is two important changes in comparison to the parser on GTT.Z5. The first one concerns the rewriting of formulas. Why that? Simple. When processing a branch, there are three things that we can do, supposing we take the (reasonable) strategy of processing non-branching formulas (like true conjunctions, false implications...) first: (i) we can go through the branch and look for a true conunction. If there is none, go again and look for a false implication. If there is none, etc. (ii) we can go through the branch and look for the first wff which is a true conjunction or a false implication or... (iii) we can rewrite wffs on parsing, eliminating all booleans but, say, conjunctions. For instance, avb is equivalent to ¬(¬a&¬b). So instead of creating a new node with v (a, b), we create a node with -&(-a, -b). It uses the same place, and we don't have to care for disjunctions anymore. By the way, I'll also be using positive number to represent true wffs, and negative numbers for false ones. Thus, finding 97 in a branch means we have Ta there, whilst finding -& means we have a false conjunction.

The side effects of this approach are two. First, we don't check anymore if some node is present. For instance, if we had a formula like a&b -> a&b, we will end up having a tree with twice the same node, as following:

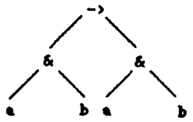

fig. 26

And it should be clear why: take the formula ¬a&a. On parsing, we will have a conjunction between a positive wff a and a negative one—we *have* to store them in different places, so there is no need to (actually one cannot) check whether some subformula is already there. And second, as I explained before, equivalences are being left out—since a<->b is equivalent to, say, ¬(a&¬b)&¬(b&¬a) one would have to copy the two subformulas somewhere else, because they occur with different signs.[52]

10.3 tableau.h

Let us now discuss how we implemente the tableau procedure. Before we dive into (parts of) the code of the tableau() function, let us talk a little while about the way things are supposed to work.

In the semantics for ZP5, we learned that, since the knowledge branch is S5, that every knowledge formula has the same value in all worlds, the same holding (witness monoclustered models) for belief formulas. So we don't have in principle to care in which world a modalized formula holds, or not. This is the reason why I introduced a new stack, MOD, which is like PRF (where we store the branch), only with modalized formulas, to begin with. But not only do modalized formulas hold in every world. Suppose we

[52] But of course one can use this copy mechanism. See the function copy() on the program in the next chapter, which could be used to accomplish this.

have a true Kp wff: then p is also true in every world. So, when we process Kp, we won't add p to PRF, but equally to MOD. The only wffs which go to PRF are those whose truth is not "universal", so to speak, or those which lead to branchings. I explain: suppose we have K(avb) true. Then avb is true in every world. But we must be careful not to confuse this with either a is true in every world, or b is true in every! What would happen if we would add avb to the MOD stack and process it there. So I decided to put wffs like avb into PRF, where we process it in every existing world, if needed. Suppose that we have the worlds 0, 1 and 2. We begin adding avb to PRF with index 2, and then store in a field in ALT that there is also 2 worlds more to be checked, if needed. If we don't get a contradiction with avb on world 2, we "backtrack" and try again, adding it to PRF with index 1. If somewhen a contradiction is found, the other alternatives don't need to be considered anymore.

Speaking of contradictions, some words about how we find one. First, we have in PRF that all wffs have an index (in the field PRF[x][2], x being a line). Thus, finding a and -a in PRF such that the index of a is equal to the index of -a means that is some world an atom is getting both truth and falsity, hence we have a contradiction. But another possibility is when we have an atom in MOD—like processing a true Kp and adding p to MOD. So, if we find a -p in MOD, or, for that matter, in PRF with any index, we also have contradiction. Or not?

Would that it were so simple. Remember, we are also putting belief wffs in MOD. To see why this constitutes a problem, suppose one has -a in world 0 (taken to be the initial one), and Ba true in MOD: a is added to MOD. Now if we say this is a contradiction, we'll end up having Ba->a true. So here is where the special array BLF comes into picture: there we store for each wff whether it has a "belief antecedent" or not. (Or, to put it more precisely, whether the reduced formula belongs to an open world, or not.) Thus, when adding a to MOD because Ba is true we make sure that we put a 1 into BLF[*location of* a]. The contradiction function also checks for this, so we won't have problems.

But let us take a look at parts of the code.

```
tableau()
{
      short gt, val, done;

      done = val = 0;
      bckt_ptr = CNULL;
```

gt is a loop variable, which we use while we are looking for a special wff (say, looking for a true conjunction); val denotes the current number of valuations (worlds), and done is there to indicate whether we are through with the tableau construction or not. They are set to 0 on the beginning, and bckt_ptr (the counter in BCKT) to CNULL.

Now we enter the Main Tableau Loop: while not done, or as long as there are complex formulas in a branch, we try to reduce them.

```
      while ( !done )
      {
            if ( CONTRADICTION )
            {
                  if ( bckt_ptr == CNULL )
                        SUCCEED
                  else
                        restore_state();
            }
```

First thing we do in a branch is to check where there is a contradiction. In this we call two functions: first, iw_contrad(), which examines whether there is an "inter_world" contradiction, that is, it begin looking the atoms in MOD. If not, we call a "normal" contradiction, contrad(), which proofs only the atoms in PRF.

Now, supposing we found a contradiction, we have to look whether there is another branch in store which we need to consider. If yes (bckt_ptr is greater than CNULL), we restore the state as it was before the branching, and go on. If not, this one was the last (or only) branch, so we are done and tableau() returns 1.

Now suppose we didn't find a contradiction. So we have to look for complex formulas which we are going to process. We first LOOK_FOR a true conjunction in MOD (this is what this macro does). Suppose we find one (that is, FOUND holds) at location gt. Now we have to examine which kind of formulas this conjunction's subformulas are. First we examine the left one: if it's modal (KB...) then we add it to MOD. If this is also true of the right one, same procedure. Now in this case we have to remove the old wff from PRF, what can be accomplished in two ways: if gt is the bottom line of the branch, we just reduce the prf_ptr by one. Else we set gt's operator to USD, to indicate there is nothing worth looking in this line.[53]

But we also have the other possibilites. If the right subformula is not a modal one, we simply replace the conjunction with it. Similar if the right one is modalized and the left not. Now in case both are not modals, we have to increase the pointer in PRF by one and put the left subformula there, replacing the conjunction with the right one.

```
LOOK_FOR( UND );
if ( FOUND )       /* a true conjunction was found */
{
          if ( KB( OP( gt, L_SUB ) ) )
          {
                    T_ADD_2_MOD( LOC( gt, L_SUB ) );
                    if ( KB( OP( gt, R_SUB ) ) )
                    {
                              T_ADD_2_MOD( LOC( gt, R_SUB ) );
                              UPDATE_PRF;
                    }
                    else
                    {
                              REPL_WITH( R_SUB );
                    }
          }
          else
          if ( KB( OP( gt, R_SUB ) ) )
          {
                    T_ADD_2_MOD( LOC( gt, R_SUB ) );
                    REPL_WITH( L_SUB );
          }
          else
          {
                    ADD_TRUE( gt, R_SUB );
                    REPL_WITH( L_SUB );
          }
}
```

[53] Working with dynamic memory allocation, e.g. with a linked list, you would just remove the wff: no "blanks" between.

If we couldn't find a true conjunction in PRF (only atoms and false conjuntions are there), the next case is LOOK_M_FOR(UND), which does the same procedure as before, only looking in MOD. (See listing on Appendix C)

Suppose now there is also no true conjunction in MOD: then we go back to PRF and try to find a false one.

```
LOOK_FOR( -UND );
if ( FOUND )        /* a false conjunction found */
{
        STORE_RIGHT_WFF;
        save_state( gt );
        if ( KB( OP( gt, L_SUB )) )
        {
                F_ADD_2_MOD( LOC( gt, L_SUB ) );
        }
        else
        {
                F_REPL_WITH( L_SUB );
        }
}
```

If we succeed with the search, we store the right subformula (which hold in the other branch) on BCKT. That means increasing bckt_ptr and also storing some information there: for instance, how many world there are, which value have the different pointers (prf_ptr, mod_ptr, alt_ptr...), which formula gave rise to this branch, which modals were still unprocessed, and so on. This is what STORE_RIGHT_WFF and save_state() accomplish; see the listing in Appendix C for the actual code. Now we check which kind of wff the left subformula is: if a modal, add to MOD, else replace the false conjunction in PRF with it.

Now the next step, if there were no false conjunctions, it's processing knowledge and belief wffs, if any. So we LOOK_4_KB(-KNW, -BEL)—first trying the false ones:

```
LOOK_4_KB( -KNW, -BEL );
if ( FOUND ) /* a false modality was found */
{
        if( MOD[gt][0] == -KNW )
        {
                MOD[gt][0] = -U_KNW;
                if ( KB( OP_M( gt, L_SUB )) )
                {
                        F_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                }
                else
                {
                        ADD_SCOPE( ++val, gt, -NEXT( prf_ptr ) );
                        if ( !KB( OP_M(gt,L_SUB)) )
                              { BLF[LOC_M(gt,L_SUB)] = 1; }
                }
        }
        else  /* == -BEL */
        {
                MOD[gt][0] = -U_BEL;
                if ( KB( OP_M( gt, L_SUB )) )
                {
                        F_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                }
                else
                {
                        ADD_SCOPE( ++val, gt, -NEXT( prf_ptr ) );
                }
        }
}
```

Having found something, we first check whether is a false knowledge or false belief wff, and act accordingly. Suppose it is a false KNW: we replace the operator with -USD_K, to show it was already processed (so we won't try it again!) We then look at the subformulas. If modal, add to modal, with the care of updating BLF if necessary. That is, the scope of this false KNW can be a closed world, so we have to indicate it. Else (left subwff is not modal), create a new world: this is done with ADD_SCOPE: val is increased, and we add to PRF the false subwff with a new index. Then ALT is updated, as I mentioned in the beginning of this section.

The case of belief is handled in a similar way.

If there were no false knowledge or beliefs, then look for true ones.

```
LOOK_4_KB( KNW, BEL );
if ( FOUND )        /* a true modal was found */
{
      if( MOD[gt][0] == KNW )
      {
            MOD[gt][0] = U_KNW;
            if ( OP_M( gt, L_SUB ) != -UND )
            {
                  T_ADD_2_MOD( LOC_M( gt, L_SUB ) );
            }
            else
            {
                  ADD_SCOPE( val, gt, NEXT( prf_ptr ) );
                  UPDATE_ALT;
            }
      }
      else  /* == BEL */
      {
            MOD[gt][0] = U_BEL;
            if ( OP_M( gt, L_SUB ) != -UND )
            {
                  T_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                  if ( !KB( OP_M( gt, L_SUB )) )
                  {
                        BLF[LOC_M(gt,L_SUB)] = 1;
                  }
            }
            else
            {
                  ADD_SCOPE( val, gt, NEXT( prf_ptr ) );
                  UPDATE_ALT;
            }
      }
}
```

First we decide whether we're dealing with KNW or BEL, and set the operator to "used". And we go to a similar song-and-dance as before, checking for what the subformula is to see where to put it and remebering to update ALT and BLF if need arises.

Now suppose that after this long search we found just nothing: no contradictions, and no more formulas to be reduced in this branch. We are on our last hope:

```
else
if ( OTHER_WORLDS )
{
      GO_NEXT_WORLD;
}
else
      done = 1;
```

OTHER_WORLDS looks whether ALT has another indices stored: if yes, go to the next alternative. This results, as I mentioned above, of having added a false conjunction which is "universally true". We have to check each world to see whether we get a contradiction there. The loop continues. If there are no more worlds to be checked, (a certain field in ALT was set to 0) we are done.

And that's it.

10.4 A working session with TTP.ZP5

Now some examples from what happens when the program runs: let us type some formulas and see what happens:

```
*****************************************************
*              TABLEAU THEOREM PROVER FOR ZP5       *
*                    Cesar A. Mortari               *
*                    V1.0, July 1990                *
*****************************************************

Syntax:
        a..t (variables), ¬, K, B, &, v, ->

To exit type ';'<CR>

Please type in a formula:
Ka->a
yes
Time: 0 ms

Ba->a
no
Time: 16 ms

a->Ka
no
Time: 16 ms

K(a->b)->(Ka->Kb)
yes
Time: 16 ms

B(a->b)->(Ba->Bb)
yes
Time: 16 ms

¬Ba->B¬Ba
yes
Time: 0 ms

¬K¬Ka->K¬K¬a
yes
Time: 16 ms

Ka->KKa
yes
Time: 16 ms

Ba->KBa
yes
Time: 0 ms
```

```
Ba->BKa
no
Time: 0 ms

¬Ka->K¬Ka
yes
Time: 0 ms

¬Ba->K¬Ba
yes
Time: 0 ms

K(avb)->KavKb
no
Time: 16 ms

KavKb->K(avb)
yes
Time: 16 ms
```

11

Implementation of the algorithm

C, n.:

A programming language that is sort of like Pascal except more like assembly except that it isn't very much like either one, or anything else. It is either the best language available to the art today, or it isn't.

RAY SIMARD

In this chapter I'll present a C program which uses the theorem prover for **ZP5** from the last Chapter in order to implement the algorithm (see chapter 2) which decides whether some formula belongs to a minimal belief state or not. The algorithm itself is not very hard to implement, once we have a theorem prover for the logic in question.

The program, which is called `ALG.ZP5`, is an extension of TTP.ZP5. The big change is that now we first type a wff, which describes everything that Angela believes, and then we are asked for another one. The program then checks whether the second formula belongs to Angela's belief state, given that she believes only the first one. The program calls the tableau procedure and returns an "yes" or "no" . The main loop does the following things:

- reads a formula (string) from the standard input;
- parses the formula;
- reads a second formula (string) from the standard input, and parses it;
- runs the algorithm we discussed on chapter 2; and
- prints the answer on the screen.

11.1 Data Structures and `main()`

The macros I added to TTP.ZP5 are only a few of minor importance. With regard to global variables, the only changes are:

```
short        WFF[STR_LEN][4],
            . . .
            . . .
            e;
```

As one can see, WFF now has another line, where we store the modal degree of the formulas. And a global e was added, because we'll have now and then to add to WFF some new formulas. Remember, the algorithm states that a formula A belongs to Angela's belief state, when everything she believes is α, if believing A is a consequence of believing α and *also of the modalized subformulas of A which have already been decided*. So it's likely that we'll end up having to store these subformulas somewhere: suppose we get some subformula Kp of A which it doesn't belong to Angela's belief state. We have to build a conjunction of Bα and $\neg$Kp, before checking whether this implies BA, and this means *copying* this Kp (now with a *minus* sign) somewhere else.

Now the main() function has a similar code as in the previous programs, but our goals here are other, so let us look at it. Basically there is again some startup information being outputed on the screen, after what one reads a wff (or ';' to end the program), and proceeds to parse it. Then a second wff is read in, and it goes on like:

```
if ( formula( 0, &u, &i ) && theWff[u] == '\0' )
{
      printf("\nNow type the next wff:\n");
      gets( theWff );
      k1 = TickCount();
      if ( formula( 0, &u, &j ) && theWff[u] == '\0' )
      {
            a_ptr = i;
            e = wf_ptr;
            WFF[e][0] = BEL;
            WFF[e][1] = a_ptr;
            p_ptr = j;
            if ( WFF[p_ptr][3] > 0 )
                  for ( s = 1; s <= WFF[p_ptr][3]; ++s )
                        loop( s );
            prf_ptr = bckt_ptr = mod_ptr = alt_ptr = CNULL;
            e = p_ptr+1;
            add_fi_psi( p_ptr );
            prepare_tab( p_ptr, 0 );
            for ( i = 0; i < STR_LEN; ++i )
                  BLF[i] = 0;
            if ( tableau()) ...
```

After having succesfully parsed the first wff, we read the second one. If it is also OK, one adds Bα to the WFF stack, and look which degree the second wff (pointed at by p_ptr) has. Starting with 1 and going until *degree*, we apply the algorithm on subwffs of the wff in p_ptr, as the algorithm prescribes.This is what loop() does (we'll discuss it afterwards). Then we add a 1 or 0 to of WFF[*subwff*][2], for each examined subformula, in order to inform whether it belongs or not to Angela's belief state. After having considered all subwffs, we run the algorithm on the main wff: we add the conjunctions of modalized subformulas (with add_fi_psi()); the initial information for the tableau proof procedure is set (that is, Bα, the wff we are examining, and so on, are added to the stacks PRF or MOD, depending on their main operator). Finally, tableau() is called, and the answer is printed.

We are now ready to consider more details of the program. The other routines are in several files, which are included (with #include <*file*>) just before main(). They are

```
#include "macros.h"
#include "prototypes.h"
#include "parser.h"
#include "tableau.h"
#include "subf.h"
```

Everything is (almost) the same; the new functions which implement the algorithm were added together in `subf.h`. A small change was made in `parser.h`: now the modal degree of a wff is automatically computed on parsing. (See listing on Apendix D.) The file `tableau.h` is the same. So let us look at what is new.

11.2 subf.h

We have three functions in this file. The first one is `loop`:

```
void loop( s ) /*
-------------- */
short s;
{
        short p, j;

        p = a_ptr+1;
        while ( p <= p_ptr )
        {
                prf_ptr = bckt_ptr = mod_ptr = alt_ptr = CNULL;
                if ( WFF[p][3] == s && KB( WFF[p][0] ) )
                {
                        add_fi_psi( WFF[p][1] );
                        prepare_tab( p, 1 );
                        WFF[p][2] = tableau();
                }
                ++p;
        }
}
```

`Loop` runs with an index `s` which denotes the degree which we are interested in. It makes a loop with index `p` from the place where our α is stored until `p_ptr`, which points to the second wff (the one we are trying to decided wheter it's in Angela's belief state). If the wff at `p` is a modalized one and its degree is equals to `s`, we add to `WFF` its already decided modal subformulas (for degree 1 this is just nothing), prepare the stacks and call `tableau()`. Then set `WFF[p][2]` to whichever value (1 or 0) `tableau()` returns. And go on until all wffs of degree `s` have been processed. Back to the main loop, `loop` is called again with `s+1`, and so on.

Let us now look at `add_fi_psi()`, which is responsible for adding to `WFF` modal subformulas (of a certain wff) which have already been decided.

```
void add_fi_psi( p ) /*
----------------*/
short p;
{
        short l, r;

        if ( ATOMIC( WFF[p][0] ))
                return;
        if ( KB( WFF[p][0] ))
        {
                r = e;
```

```
            l = copy( WFF[p][1] );
            ++e;
            if ( WFF[p][2] == 1 )
                 WFF[e][0] = PST( WFF[p][0] );
            else
                 WFF[e][0] = NGT( WFF[p][0] );
            WFF[e][1] = l;
            ++e;
            WFF[e][0] = UND;
            WFF[e][1] = r;
            WFF[e][2] = e-1;
            add_fi_psi( WFF[p][1] );
      }
      else  /* conjunction */
      {
            add_fi_psi( WFF[p][1] );
            add_fi_psi( WFF[p][2] );
      }
}
```

If the wff p points at is atomic, it has obviously no modal subwffs, so we exit. Else, if we are considering some knowledge of belief formula, make a copy of it in the WFF array (we have to work with copies, of course). And call add_fi_psi() on the subformulas. If the formula being considered is a conjunction, call add_fi_psi()on its subformulas.

The other two function sin subf.h are, first, copy(), which just does what its name says. (Why this is needed was already discussed. See its coding on Appendix C.) And the other is prepare_tab(), which just add wffs to the correct arrays (PRF or MOD) before calling tableau().

And that's it.

11.3 A working session with ALG.ZP5

Now some examples from what happens when the program runs: let us type some formulas and see what happens:

```
****************************************************
*              BELIEF STATE ALGORITHM FOR ZP5       *
*                    Cesar A. Mortari               *
*                     V1.0, August 1990             *
****************************************************

Syntax:
        a..t (variables), ¬, K, B, &, v, ->, <->

To exit type ';'<CR>

Please type in your 'alpha':
a

Now type the next wff:
Ba
yes
Time: 16 ms

Please type in your 'alpha':
a&b
```

```
Now type the next wff:
Bc
no
Time: 16 ms

Please type in your 'alpha':
a&b

Now type the next wff:
¬Bc
yes
Time: 16 ms


Please type in your 'alpha':
a&b

Now type the next wff:
B¬Bc
yes
Time: 16 ms

Please type in your 'alpha':
a&b

Now type the next wff:
KBa
yes
Time: 16 ms


Please type in your 'alpha':
a&b

Now type the next wff:
K¬Kc
yes
Time: 16 ms


Please type in your 'alpha':
¬Kc

Now type the next wff:
Kc
no
Time: 16 ms


Please type in your 'alpha':
¬Kc

Now type the next wff:
B¬Kc
yes
Time: 16 ms

Please type in your 'alpha':
avb

Now type the next wff:
Ba
no
Time: 33 ms


Please type in your 'alpha':
avb
```

```
Now type the next wff:
Bb
no
Time: 16  ms


Please type in your 'alpha':
avb

Now type the next wff:
¬Ba
yes
Time: 16  ms


Please type in your 'alpha':
avb

Now type the next wff:
¬Bb
yes
Time: 16  ms

Please type in your 'alpha':
avb

Now type the next wff:
B(avb)
yes
Time: 16  ms
```

Final remarks

A conclusion is simply the place where someone got tired of thinking.

A few words to conclude this work. We set out with the goal of characterizing belief states in cases where agents have only partial information about some domain. In the course of the investigation, we saw that there were different alternatives which enabled us to reach this goal. Not every one of them worked, or worked equally well, for every logic, but a nice feature was that we obtained an algorithm with which to characterize the belief states. This lead us to the other two parts of this work, which dealt, first, with decision procedures via valuation semantics for modal and epistemic logics, and second, with implementations of these procedures.

What could we now say more about our original goal? As the old saying goes, every solution immediately raises more questions than we had former. So it is not surprising that this should be the case here too. So let me mention some open problems, or rather, some directions to further investigations.

First, it is still open whether one can find a reasonable characterization method for the logics **Z**, **ZG** and **ZP** via trying to find the smallest stable set. Chances are small, because of the infinite number of modalities, but it would be nice to have a definite answer.

Second, one should also investigate whether formulas like $B(\neg KA \rightarrow B)$ really behave like default rules, which was one of the motivations to use logics of knowledge and belief. I left this question untouched, because a thorough research on it would constitute by itself another dissertation. Supposing thus we have an affirmative answer, one could then investigate, with respect to the different EDL-systems, the resulting default logics.

Third, until now, as we have seen, I have kept to the case in which we consider only one agent, but the most interesting situations would be of course the ones involving more, interacting agents—as, for instance, in a distributed system. A strong suspicion, not to say certainty, is that things will be a lot more complicated, witness Halpern and Moses' remarks with respect to a multi-agent, S5-based knowledge logic.

And fourth, what happens if agents are not ideal, either not logically omniscient, or not fully introspective? Working on this problem presupposes first of all the existence of EDLogics with respect to which agents have these desired characteristics, and, as I had opportunity to mention, we are far from having, for instance, reasonable non-omniscient logical systems. So an important direction for further research is the development of "more realistic" logics of knowledge and belief—a topic which particularly interests me, and which I pretend to consider in future works.

Appendices
and
References

A

Some derivations

One should not clutter one's mind with trivialities.

G. Harman, *Change in View*

A1. **Z5 and Z5* are the same logic:**

($Z5 \subseteq Z5^*$)

1. $KKA \to BKA$	m
2. $KA \to KKA$	4
3. $KA \to BKA$	$1,2\ TR$

($Z5^* \subseteq Z5$)

1. $KA \to A$	t
2. $BKA \to BA$	$1,\ RB,\ k^b$
3. $KA \to BKA$	m^*
4. $KA \to BA$	$2,3\ TR$

A2. ***p* and *p** are equivalent:**

($p \Rightarrow p^*$)

1. $BBA \to \neg B \neg BA$	d^b
2. $B \neg BA \to \neg BBA$	$1,\ Transp$
3. $KB \neg BA \to K \neg BBA$	$2,\ RK,\ k$
4. $B \neg BA \to KB \neg BA$	p
5. $\neg BA \to B \neg BA$	5^b
6. $\neg BA \to KB \neg BA$	$4,5\ TR$
7. $BA \to BBA$	4^b
8. $BA \to \neg B \neg BA$	$1,7\ TR$
9. $B \neg BA \to \neg BA$	$8,\ Transp,\ DN$
10. $KB \neg BA \to K \neg BA$	$9,\ RK,\ k$
11. $\neg BA \to K \neg BA$	$6,10\ TR$

$(p^* \Rightarrow p)$

1. $BA \rightarrow BBA$	4^b
2. $BBA \rightarrow \neg B \neg BA$	d^b
3. $BA \rightarrow \neg B \neg BA$	*1,2 TR*
4. $\neg B \neg BA \rightarrow K \neg B \neg BA$	p^*
5. $BA \rightarrow K \neg B \neg BA$	*3,4 TR*
6. $\neg BA \rightarrow B \neg BA$	5^b
7. $\neg B \neg BA \rightarrow BA$	*6, Transp*
8. $K \neg B \neg BA \rightarrow KBA$	*7, RK, k*
9. $BA \rightarrow KBA$	*5,8 TR*

A3. $\vdash_{ZCP} BA \leftrightarrow \neg K \neg KA$

1. $KA \rightarrow BA$	*m*
2. $\neg BA \rightarrow \neg KA$	*1, Transp*
3. $K(\neg BA \rightarrow \neg KA) \rightarrow (K \neg BA \rightarrow K \neg KA)$	*k*
4. $K(\neg BA \rightarrow \neg KA)$	*2, RK*
5. $K \neg BA \rightarrow K \neg KA$	*3,4 MP*
6. $\neg K \neg KA \rightarrow \neg K \neg BA$	*5, Transp*
7. $\neg BA \rightarrow K \neg BA$	p^*
8. $\neg K \neg BA \rightarrow BA$	*7, Transp, DN*
9. $\neg K \neg KA \rightarrow BA$	*6,8 TR*
10. $K \neg KA \rightarrow B \neg KA$	*m*
11. $\neg B \neg KA \rightarrow \neg K \neg KA$	*10, Transp*
12. $BKA \rightarrow \neg B \neg KA$	d^b
13. $BKA \rightarrow \neg K \neg KA$	*11,12 TR*
14. $BA \rightarrow BKA$	*c*
15. $BA \rightarrow \neg K \neg KA$	*13,14 TR*
16. $BA \leftrightarrow \neg K \neg KA$	*9,15, Df*$\leftrightarrow$

A4. $\vdash_{ZCP} B(BA \rightarrow KA)$

1. $B(KA \rightarrow (BA \rightarrow KA)) \rightarrow (BKA \rightarrow B(BA \rightarrow KA))$	k^b
2. $B(KA \rightarrow (BA \rightarrow KA))$	*Taut, RB*
3. $BKA \rightarrow B(BA \rightarrow KA)$	*1,2 MP*
4. $BA \rightarrow BKA$	*c*
5. $BA \rightarrow B(BA \rightarrow KA)$	*3,4 TR*
6. $B(\neg BA \rightarrow (BA \rightarrow KA)) \rightarrow (B \neg BA \rightarrow B(BA \rightarrow KA))$	k^b
7. $B(\neg BA \rightarrow (BA \rightarrow KA))$	*Taut, RB*
8. $B \neg BA \rightarrow B(BA \rightarrow KA)$	*6,7 MP*
9. $\neg BA \rightarrow B \neg BA$	5^b
10. $\neg BA \rightarrow B(BA \rightarrow KA)$	*8,9 TR*
11. $BA \vee \neg BA \rightarrow B(BA \rightarrow KA)$	*5,10 Taut*
12. $B(BA \rightarrow KA)$	*11, Taut, MP*

B

GTT.Z5 Listings

Real programmers don't comment their code. It was hard to write, it should be hard to understand.

B1. File: macros.h

```
/*      -------------- MACROS -------------------       */

#define STR_LEN             256          /* length of strings       */
#define MAX_LINES           64           /* max nr. of lines in a table */
#define MAX_ROWS            30           /* max nr. of rows in a table */
#define CNULL               (-1)         /* a "pointer" to NIL      */
#define UND                 5            /* conjunction */
#define ODR                 6            /* disjunction */
#define IMP                 7            /* implication */
#define NEG                 8            /* negation */
#define EQU                 9            /* biconditional */
#define KNW                 10           /* knowledge */
#define BEL                 11           /* belief */

#define P_ATOMIC( c )       (c >= 'a' && c <= 't')  /* atoms on input */
#define ATOMIC( c )         (c > BEL)    /* internally, atoms are
                                         everything but connectives   */

#define INIT_NODE(a,b,c)    WFF[wf_ptr][0] = a; WFF[wf_ptr][1] = b; WFF[wf_ptr][2] = c


/* -------------- GLOBALS   -----------------   */

short   WFF[STR_LEN][3],            /* stores the tree representation of the wff */
        wf_ptr,                     /* pointer to current location in WFF      */
        TBL[MAX_ROWS][MAX_LINES];   /* the GTT */
char    theWff[STR_LEN];            /* input formula */
long    k1,k2,k3,k4;                /* for time measurement */
```

B2. File: prototypes.h

```
/*      -------------- FUNCTION PROTOTYPES ---------------- */

/* -- parser.h -- */

int       formula( short, short *, short * );
int       form_and_or( short, short *, short * );
int       rest_fd( short, short, short *, short * );
int       rest_form( short, short, short *, short * );
int       form1( short, short *, short * );
short     get_place( short );
int       unify ( short, short );

/* -- table.h -- */

short      make_table( void );
void       copy_lines( short, short );
void       print_table( short, short );
void       displayWFF( void );
void       copy_lines( short, short );
int        alphaNE( short, short, short, short, short, short );
int        gammeNE( short, short, short, short, short );
int        accesK( short, short, short, short );
int        accesB( short, short, short, short );
int        deltasNE(  short, short, short, short, short );
int        betasNE(  short, short, short, short, short );
int        gammaP( short, short, short, short, short, short );
int        alphaP( short, short, short, short, short, short );
void       split_lines( short, short, short );
```

B3. File: parser.h

```
/* ------------------------ PARSER FUNCTIONS ------------------------ */
/*
       The parser -- formula() -- processes the input string and store it
       if a wff) in the WFF array, in a treelike representation. This parser
       is based in a Prolog one developed by Franz Guenthner. The overall
       structure is the same, but of course we adapted it to our data
       structures here.
*/

formula( xi, xo, z ) /*
-------------------- */
short  xi, *xo, *z;
{
       short  xn, zn;

       if ( form_and_or( xi, &xn, &zn ) )
       {
              if ( rest_form( xn, zn, xo, z ) )
                     return 1;
              else
                     return 0;
       }
       else
              return 0;
}

form_and_or( xi, xo, zo ) /*
------------------------- */
short  xi, *xo, *zo;
{
       short  zi, xn;

       if ( form_1( xi, &xn, &zi ) )
       {
              if ( rest_fd( xn, zi, xo, zo ) )
                     return 1;
       }
       else
              return 0;
}

rest_fd( xi, zi, xo, zo ) /*
------------------------- */
short  xi, zi, *xo, *zo;
{
       short  z, i, old_ptr;

       old_ptr = wf_ptr;

       if ( theWff[xi] == '&' && form_and_or( xi+1, xo, &z ))
```

```
        {
                INIT_NODE( UND, zi, z );
                i = get_place( wf_ptr-1 );
                if ( i == CNULL )
                        *zo = wf_ptr++;
                else
                {
                        *zo = i;
                        wf_ptr = old_ptr;
                }
        return 1;
    }
        else
        if ( theWff[xi] == 'v' && form_and_or( xi+1, xo, &z ))
        {
                INIT_NODE( ODR, zi, z);
                i = get_place( wf_ptr-1  );
                if ( i == CNULL )
                        *zo = wf_ptr++;
                else
                {
                        *zo = i;
                        wf_ptr = old_ptr;
                }
        return 1;
    }
        else
        {
                *xo = xi;
                *zo = zi;
        return 1;
        }
}

rest_form( xi, zi, xo, zo ) /*
---------------------------- */
short   xi, zi, *xo, *zo;
{
        short  z, old_ptr, i;

        old_ptr = wf_ptr;

        if (( theWff[xi] == '-' ) && ( theWff[xi+1] == '>' )
                        &&  formula( xi+2, xo, &z ))
        {
                INIT_NODE( IMP, zi, z );
                i = get_place( wf_ptr-1 );
                        if ( i == CNULL )
                                *zo = wf_ptr++;
                        else
                        {
                                *zo = i;
                                wf_ptr = old_ptr;
                        }
                return 1;
        }
        else
        if (( theWff[xi] == '<' ) && ( theWff[xi+1] == '-' )
                        && ( theWff[xi+2] == '>' ) && formula( xi+3, xo, &z ))
        {
                INIT_NODE( EQU, zi, z );
                i = get_place( wf_ptr-1  );
                if ( i == CNULL )
                        *zo = wf_ptr++;
                else
                {
                        *zo = i;
                        wf_ptr = old_ptr;
                }
                return 1;
        }
        else
        {
                *xo = xi;
                *zo = zi;
                return 1;
        }
}

form_1( xi, xo, zo ) /*
------------------ */
short   xi, *xo, *zo;
{
        short  f, i, old_ptr;

        old_ptr = wf_ptr;
        if ( (unsigned char) theWff[xi] == (unsigned char) '¬' )
        {
                if ( form_1( xi+1, xo, &f ))
                {
                        INIT_NODE( NEG, f, CNULL);
                        i = get_place( wf_ptr-1  );
                        if ( i == CNULL )
                                *zo = wf_ptr++;
                        else
                        {
                                *zo = i;
                                wf_ptr = old_ptr;
                        }
                        return 1;
                }
                else
                        return 0;
        }
        if ( theWff[xi] == 'K')
        {
                if (form_1( xi+1, xo, &f ))
                {
                        INIT_NODE( KNW, f, CNULL);
                        i = get_place( wf_ptr-1  );
                        if ( i == CNULL )
                                *zo = wf_ptr++;
                        else
                        {
                                *zo = i;
                                wf_ptr = old_ptr;
                        }
                        return(1);
                }
                else
                        return(0);
        }
        if ( theWff[xi] == 'B')
        {
                if (form_1( xi+1, xo, &f ))
                {
                        INIT_NODE( BEL, f, CNULL);
                        i = get_place( wf_ptr-1  );
                        if ( i == CNULL )
                                *zo = wf_ptr++;
                        else
                        {
                                *zo = i;
                                wf_ptr = old_ptr;
                        }
                        return(1);
                }
                else
                        return(0);
        }
        if ( theWff[xi] == '(' )
        {
                if ( formula( xi+1, xo, &f ))
                {
                        if ( theWff[*xo] == ')' )
                        {
                                ++(*xo);
                                *zo = f;
                        return 1;
                }
                        else
                                return 0;
                }
        }
        if ( P_ATOMIC( theWff[xi]) )
        {
                i = wf_ptr-1;
                while ( i != CNULL && WFF[i][0] != theWff[xi] )
                        --i;
                if ( i == CNULL )
                {
                        WFF[wf_ptr][0] = theWff[xi];
                        *zo = wf_ptr++;
                }
                else
                        *zo = i;
                *xo = xi+1;
            return 1;
    }
        else
                return 0;
}

short   get_place( ptr ) /*
--------------------------*/
short   ptr;
{

        while ( ptr != CNULL && WFF[ptr][0] != WFF[wf_ptr][0] )
                --ptr;
        if ( ptr == CNULL )
                return ptr;
        if ( unify( WFF[ptr][1], WFF[wf_ptr][1] ))
        {
                if ( WFF[WFF[ptr][1]][0] == NEG )
                        return 1;
                if ( unify( WFF[ptr][2], WFF[wf_ptr][2] ))
                        return ptr;
                else
                        return( get_place( ptr-1 ));
        }
        else
                return( get_place( ptr-1 ));
}

unify( x, y ) /*
------------- */
short   x, y;
{
        if ( WFF[x][0] != WFF[y][0] )
                return 0;
        else
        {
                if ( ATOMIC( WFF[x][0] ))
                        return 1;
                else
                {
                        if ( !unify( WFF[x][1], WFF[y][1] ) )
                                return 0;
                        else
                        {
                                if ( unify( WFF[x][2], WFF[y][2] ) )
                                        return 1;
                                else
                                        return 0;
                        }
                }
        }
}
```

B4. File: outputs.h

```
/*      ----- Functions for printing results ----- */

void displayWFF() /*            -- shows the contents of WFF --
------------------*/
```

```
{
        short   j;

        printf("\n");
        for ( j = 0; j < wf_ptr; ++j )
        {
                switch( WFF[j][0] )
                {
                        case NEG :
                        putchar('[');putchar('-');putchar(']');
                        break;
                        case KNW :
                        putchar('[');putchar('K');putchar(']');
                        break;
                        case BEL :
                        putchar('[');putchar('B');putchar(']');
                        break;
                        case UND:
                        putchar('[');putchar('&');putchar(']');
                        break;
                        case ODR:
                        putchar('[');putchar('v');putchar(']');
                        break;
                        case IMP:
                        putchar('[');putchar('>');putchar(']');
                        break;
                        case EQU:
                        putchar('[');putchar('=');putchar(']');
                        break;
                        default:
                        putchar('[');putchar( WFF[j][0] );putchar(']');
                }
        }
        printf("\n");
        for ( j = 0; j < wf_ptr; ++j )
        {
                switch( WFF[j][0] )
                {
                        case NEG:
                        case KNW:
                        case BEL:
                        case UND:
                        case ODR:
                        case IMP:
                        case EQU:
                        printf("[%d]",WFF[j][1]);
                        break;
                        default:
                        putchar('[');putchar(' ');putchar(']');
                }
        }
        printf("\n");
        for ( j = 0; j < wf_ptr; ++j )
        {
                switch( WFF[j][0] )
                {
                        case UND:
                        case ODR:
                        case IMP:
                        case EQU:
                        printf("[%d]",WFF[j][2]);
                        break;
                        default:
                        putchar('[');putchar(' ');putchar(']');
                }
        }
}

void print_table( rows, lines ) /*
------------------------------- */
short   rows, lines;
{
        short   i, j;

        printf("\n\n*** TABLE ***\n\n");
        for ( j = 0; j < rows; ++j )
```

```
                printf("[%d]", j);
        printf("\n");
        for ( j = 0; j < rows; ++j )
        {
                putchar('-');putchar('-');putchar('-');
        }
        printf("\n");
        for ( i = 0; i < lines; ++i )
        {
                for ( j = 0; j < rows; ++j )
                        printf("|%d|", TBL[j][i]);
                printf("\n");
        }
        printf("\n");
}
```

B5. File: table.h

```
short make_table() /*
------------------*/
{
        int             i,                      /* run variable */
                        row,                    /* current row in the table */
                        line,                   /* current line in the table */
                        ad_lines;               /* in the modal operator cases, nr. of lines
                                                                which were added */

        TBL[0][0] = 1;          /* initialize first row, two lines (prop var) */
        TBL[0][1] = 0;
        line = 1;
        ad_lines = 0;

        /* MAIN TABLE LOOP */
        for ( row = 1; row <= wf_ptr-1; ++row )
        {
                if ( row >= MAX_ROWS || line >= MAX_LINES )
                {
                        printf("\n\n***** ERROR : TABLE TOO LARGE! *******");
                        return 0;
                }
                switch( WFF[row][0] )                   /* switch the connectives */
                {
                        case NEG :
                                for ( i = 0; i <= line; ++i )
                                        TBL[row][i] = !TBL[WFF[row][1]][i];
                                break;
                        case UND :
                                for ( i = 0; i <= line; ++i )
                                        TBL[row][i] = TBL[WFF[row][1]][i] && TBL[WFF[row][2]][i];
                                break;
                        case ODR :
                                for ( i = 0; i <= line; ++i )
                                        TBL[row][i] = TBL[WFF[row][1]][i] || TBL[WFF[row][2]][i];
                                break;
                        case IMP :
                                for ( i = 0; i <= line; ++i )
                                        TBL[row][i] = !TBL[WFF[row][1]][i] || TBL[WFF[row][2]][i];
                                break;
                        case EQU :
                                for ( i = 0; i <= line; ++i )
                                        TBL[row][i] = (TBL[WFF[row][1]][i] == TBL[WFF[row][2]][i]);
                                break;
                        case KNW :
                        {
                                for ( i = 0; i <= line; ++i )
                                        if ( TBL[WFF[row][1]][i] == 0 )
                                                TBL[row][i] = 0;
                                        else
                                        {
                                                if ( gammaNE( row, line, WFF[row][1], 0, i ) )
                                                {
                                                        if ( deltasNE( row, line, WFF[row][1], 0, i ) )
```

```
                            {
                                split_lines( row, line+1+ad_lines, i );
                                ++ad_lines;
                            }
                            else
                                TBL[row][i] = 0;
                        }
                        else
                            TBL[row][i] = 1;
                    }
                line = line + ad_lines;
                ad_lines = 0;
                break;
            }
            case BEL :
            {
                for ( i = 0; i <= line; ++i )
                    if ( !alphaNE( row, line, WFF[row][i], 0, i, 1 ) )
                        TBL[row][i] = 0;
                    else
                    {
                        if ( alphaNE( row, line, WFF[row][i], 0, i, 0 ) )
                        {
                            if ( betasNE( row, line, WFF[row][i], 0, i ) )
                            {
                                split_lines( row, line+1+ad_lines, i );
                                ++ad_lines;
                            }
                            else
                                TBL[row][i] = 0;
                        }
                        else
                            TBL[row][i] = 1;
                    }
                line = line + ad_lines;
                ad_lines = 0;
                break;
            }
            default :               /* Atomic propositions */
            {
                copy_lines( row, line + 1 );
                for ( i = 0; i <= line; ++i )
                    TBL[row][i] = 1;
                line = 2 * (line + 1) - 1;
                while ( i <= line )
                    TBL[row][i++] = 0;
            }
        }
    }
    return ((short) line+1);
}

void    copy_lines( row, lines ) /*
--------------------------------*/
short   row, lines;
{
    short   i, j;

    for ( i = 0; i < lines; ++i )
        for ( j = 0; j < row; ++j )
            TBL[j][i + lines] = TBL[j][i];
}


alphaNE( r, l, am, b, currLine, value ) /*
-------------------------------------- */
short   r, l, am, b, currLine, value;
{
    while ( b <= l && TBL[am][b] != value )
        ++b;
    if ( b > l )
        return 0;                   /* else we're at a line with am=value */
    if ( accesB( r, l, currLine, b ))
        return 1;
    else
        return( alphaNE( r, l, am, b+1, currLine, value ));
```

```
}

gammaNE( r, l, am, b, currLine ) /*
--------------------------------*/
short   r, l, am, b, currLine;
{
        while ( b <= l && TBL[am][b] != 0 ) /* b is where to begin the search */
                ++b;
        if ( b > l )
                return 0;                               /* else we're at a line with am=0 */
        if ( accesK( r, i, currLine, b ))       /* b satisf scope of cur_line */
                return 1;
        else
                return( gammaNE( r, l, am, b+1, currLine ));
}

accesK( r, l, v, vn ) /*        -- that is, v<k,r>vn --
---------------------*/
short   r, l, v, vn;
{
        short   i;

        i = 0;
        while ( i < r && ( WFF[i][0] != KNW || TBL[i][v] == TBL[i][vn] )
                        && ( WFF[i][0] != BEL || TBL[WFF[i][1]][vn] == 1 ))
                ++i;
        if ( i >= r )
                return 1;
        else
                return 0;
}

accesB( r, l, v, vn ) /*        -- that is, v<b,r>vn --
--------------------- */
short   r, l, v, vn;
{
        short   i;

        i = 0;
        while ( i < r && ( WFF[i][0] != BEL || ( TBL[i][v] == TBL[i][vn]
                                && ( TBL[i][v] != 1 || TBL[WFF[i][1]][vn] == 1 ) ))
                                && ( WFF[i][0] != KNW || ( TBL[i][v] == TBL[i][vn] )) )
                ++i;
        if ( i >= r )
                return 1;
        else
                return 0;
}

deltasNE( r, l, am, b, v ) /*
-------------------------- */
short   r, l, am, b, v;
{
        short   i;

        i = 0;
        while ( i < r && ( WFF[i][0] != KNW || TBL[i][v] == 1 ||
                                gammaP( r, l, WFF[i][1], 0, v, am ) ))
                ++i;
        if ( i >= r )
                return 1;
        else
                return 0;

}

betasNE( r, l, am, b, v ) /*
------------------------- */
short   r, l, am, b, v;
{
        short   i;

        i = 0;
        while ( i < r && ( WFF[i][0] != BEL || TBL[i][v] == 1 ||
                                alphaP( r, l, WFF[i][1], 0, v, am ) ))
                ++i;
        if ( l >= r )
                return 1;
        else
                return 0;

}

gammaP( r, l, aq, b, v, am ) /*
-------------------------- */
short   r, l, aq, b, v, am;
{
        while ( b <= l && TBL[aq][b] != 0 )
                ++b;
        if ( b > l )
                return 0;       /* else we're at a line with aq=0 */
        if ( TBL[am][b] == 1 && accesK( r, l, v, b ))
                return 1;
        else
                return( gammaP( r, l, aq, b+1, v, am ));
}

alphaP( r, l, aq, b, v, am ) /*
-------------------------- */
short   r, l, aq, b, v, am;
{
        while ( b <= l && TBL[aq][b] != 0 )
                ++b;
        if ( b > l )
                return 0;       /* else we're at a line with aq=0 */
        if ( TBL[am][b] == 1 && accesB( r, l, v, b ))
                return 1;
        else
                return( alphaP( r, l, aq, b+1, v, am ));
}

void split_lines( r, l, v ) /*
------------------------- */
short   r, l, v;
{
        short   j;

        for ( j = 0; j < r; ++j )
                TBL[j][l] = TBL[j][v];
        TBL[r][v] = 1;
        TBL[r][l] = 0;
}
```

B6. File: main.c

```
/*****************************************************************
*                                                                *
*                     GTT.Z5 -- Version 1.0                      *
*                     Cesar A. Mortari -- May 1990               *
*                                                                *
*       This program implements the construction of a generalized *
*       truth-table for the epistemic-doxastic logic Z5.          *
*                                                                *
*****************************************************************/


/* ------------ GTT INCLUDES ------------ */

#include "macros.h"
#include "prototypes.h"
#include "parser.h"
#include "outputs.h"
#include "table.h"


main()
{
        short   i, j, k, u;
        printf("\n******************************************************");
        printf("\n*                   GTT BUILDER FOR Z5               *");
        printf("\n*                    Cesar A. Mortari                *");
        printf("\n*                     V1.0, May 1990                 *");
        printf("\n******************************************************");
        printf("\n\nSyntax:\n        a..t (variables), ¬, K, B, &, v, ->, <->\n");
        printf("\nTo exit type ';'<CR>\n");
        printf("\nPlease type in a formula:\n");
        for (;;)
        {
                /* Some initializations... */

                wf_ptr = 0;
                WFF[0][0] = CNULL;              /* to take care of first input atom...*/
                gets( theWff );                 /* input the formula to be (dis)proven */
                if ( theWff[0] == ';')          /* program ends... */
                        return 1 ;

                /* ELSE remove blanks from theWff */
                i = 0;
                j = 0;
                while ( theWff[i] != '\0' )
                {
                        if ( theWff[i] != ' ' )
                                theWff[j++] = theWff[i];
                        ++i;
                }
                theWff[j] = '\0';

                k1 = TickCount();                       /* initialize time counting */
                if ( formula( 0, &u, &i ) && theWff[u] == '\0' )
                {
                        k2 = TickCount();               /* time use in parsing */
                        k = make_table();
                        k3 = TickCount();               /* make_table time */
                        displayWFF();                   /* print the contents of WFF */
                        print_table( wf_ptr, k );
                        k4 = (( TickCount() - k3 ) * 100) / 6;          /* printing time */
                        k3 = ((k3-k2)*100)/6;
                        k2 = ((k2-k1)*100)/6;
                        printf("Parsing time: %ld ms", k2);
                        printf("\tMake table: %ld ms",k3);
                        printf("\nOutput time: %ld ms",k4);
                        printf("\tTotal time: %ld ms\n\n",k2+k3+k4);
                }
                else            /* it was not a wff, so try again... */
                        printf("\nSyntax error...\n\n");
        }
}
```

C

TTP.ZP5 Listings

Lubarsky's Law of Cybernetic Enthomology:

There is always one more bug.

C1. File: macros.h

```
/*      --------------- MACROS ------------------         */


#define STR_LEN       256          /* length of strings */

#define CNULL         (-1)         /* a "pointer" to NIL       */
#define EMPTY         (-2)
#define LMK           (-3)         /* left marker */
#define RMK           (-4)         /* right marker */
#define L_SUB         1
#define R_SUB         2
#define EQU           5            /* biconditional */
#define UND           6            /* negation */
#define KNW           7            /* knowledge */
#define BEL           8            /* belief */
#define U_KNW         9            /* knowledge, used */
#define U_BEL         10           /* belief, used */
#define USD           11


#define P_ATOMIC( c )        ( c >= 'a' && c <= 't' )
#define ATOMIC( c )          ( c > USD || c < -USD )

#define NEXT( i )                  WFF[PRF[i][1]][0]
#define NEXT_M( i )                WFF[MOD[i][1]][0]
#define OP( i, x )                 WFF[WFF[PRF[i][1]][x]][0]
#define OP_M( i, x )               WFF[WFF[MOD[i][1]][x]][0]

#define LOC( i, x )                WFF[PRF[i][1]][x]
#define LOC_M( i, x )              WFF[MOD[i][1]][x]

#define KB( x )                    ( x == KNW || x == -KNW || x == BEL || x == -BEL )

#define INIT_NODE(a,b,c)           WFF[wf_ptr][0] = a;          \
                                   WFF[wf_ptr][1] = b;    \
                                   WFF[wf_ptr][2] = c

#define F_MOD( x )                 ( MOD[x][0] == -KNW || MOD[x][0] == -BEL )
#define T_MOD( x )                 ( MOD[x][0] == KNW || MOD[x][0] == BEL )

#define SUCCEED                    ( done = 1; return 1; )

#define LOOK_FOR( x )              gt = prf_ptr;         \
                                    while ( gt != CNULL && PRF[gt][0] != x )    \
                                          —gt

#define LOOK_M_FOR( x )             gt = mod_ptr;            \
                                    while ( gt != CNULL && MOD[gt][0] != x )    \
                                          —gt

#define LOOK_4_KB( x, y )     gt = mod_ptr; \
                              while ( gt != CNULL && MOD[gt][0] != x && MOD[gt][0] != y )       \
                                    —gt

#define FOUND          gt != CNULL

#define ADD_TRUE( i, x )      ++prf_ptr;              \
                              PRF[prf_ptr][2] = PRF[i][2];       \
                              PRF[prf_ptr][1] = LOC( i, x );     \
                              PRF[prf_ptr][0] = NEXT( prf_ptr )

#define REPL_WITH( x )        PRF[gt][1] = LOC( gt, x ); \
                              PRF[gt][0] = NEXT( gt )

#define F_REPL_WITH( x )      PRF[gt][1] = LOC( gt, x ); \
                              PRF[gt][0] = -NEXT( gt )

#define ADD_MODAL( i, x )     ++prf_ptr;              \
                              PRF[prf_ptr][2] = 0;  \
                              PRF[prf_ptr][1] = LOC_M( i, x );    \
                              PRF[prf_ptr][0] = NEXT( prf_ptr )

#define ADD_SCOPE( a, l, o )        ++prf_ptr;            \
                                    PRF[prf_ptr][2] = a;  \
                                    PRF[prf_ptr][1] = LOC_M( i, l );     \
                                    PRF[prf_ptr][0] = o

#define T_ADD_2_MOD( x )            ++mod_ptr;     \
                                    MOD[mod_ptr][1] = x;  \
                                    MOD[mod_ptr][0] = NEXT_M( mod_ptr )

#define F_ADD_2_MOD( x )            ++mod_ptr;     \
                                    MOD[mod_ptr][1] = x;  \
                                    MOD[mod_ptr][0] = -NEXT_M( mod_ptr )

#define TM_REPL_WITH( l, x )        MOD[l][1] = LOC_M( l, x );   \
                                    MOD[l][0] = NEXT_M( l );

#define STORE_RIGHT_WFF             ++bckt_ptr;            \
                                    if ( KB( OP( gt, R_SUB )) )          \
                                          BCKT[bckt_ptr][3] = KNW;             \
                                    else                                       \
                                          BCKT[bckt_ptr][3] = PRF[gt][2];      \
                                    BCKT[bckt_ptr][0] = gt;       \
                                    BCKT[bckt_ptr][2] = WFF[PRF[gt][1]][2];            \
                                    BCKT[bckt_ptr][1] = -WFF[BCKT[bckt_ptr][2]][0]


#define UPDATE_PRF                  if ( gt == prf_ptr )  --prf_ptr;    \
                                    else PRF[gt][0] = USD

#define UPDATE_ALT                  ALT[++alt_ptr] = val;         \
                                    ALT[++alt_ptr] = prf_ptr-1;   \
                                    ALT[++alt_ptr] = mod_ptr;     \
                                    ALT[++alt_ptr] = bckt_ptr;    \
                                    ALT[++alt_ptr] = prf_ptr;     \
                                    ALT[++alt_ptr] = PRF[prf_ptr][1];\
                                    ALT[++alt_ptr] = PRF[prf_ptr][0]

#define OTHER_WORLDS          ALT[alt_ptr-6] > 0

#define GO_NEXT_WORLD               --ALT[alt_ptr-6];      \
                                    prf_ptr = ALT[alt_ptr-5];     \
                                    mod_ptr = ALT[alt_ptr-4];     \
                                    bckt_ptr = ALT[alt_ptr-3];   \
                                    PRF[ALT[alt_ptr-2]][0] = ALT[alt_ptr-1];     \
                                    PRF[ALT[alt_ptr-2]][1] = ALT[alt_ptr];       \
                                    PRF[ALT[alt_ptr-2]][2] = ALT[alt_ptr-6]
```

```
#define CONTRADICTION        iw_contrad( mod_ptr, prf_ptr ) || contrad( prf_ptr )

#define NON_CONTRAD   p2 != CNULL && ( PRF[p2][0] + PRF[p][0] != 0 || PRF[p2][2] != PRF[p][2] )


/* -------------  GLOBALS   ----------------   */

short  WFF[STR_LEN][3],             /* stores the tree representation of the wff */
            PRF[STR_LEN][3],
            ALT[STR_LEN],
            BLF[STR_LEN],
            BCKT[STR_LEN][4],       /* where to save states in branching */
            MOD[STR_LEN][2],
            wf_ptr,                 /* pointer to current location in WFF      */
            prf_ptr,
            mod_ptr,
            alt_ptr,
            bckt_ptr;

char   theWff[STR_LEN];             /* input formula */

long   k1, k2;                      /* for time measurement */
```

C2. File: prototypes.h

```
/*     ------------- FUNCTION PROTOTYPES -------------- */

/*     --- parser.h ----- */

int    formula( short, short *, short * );
int    form_and_or( short, short *, short * );
int    rest_fd( short, short, short *, short * );
int    rest_form( short, short, short *, short * );
int    form1( short, short *, short * );

/*     --- tableau.h ----- */

int    tableau( void );
int    contrad( short );
int    iw_contrad( short, short );
void   save_state( short );
void   restore_state( void );

/*     --- main.c ----- */

int    main( void );
```

C3. File: parser.h

```
/* ----------------------- PARSER FUNCTIONS ---------------------- */

formula( xi, xo, z ) /*
------------------- */
short  xi, *xo, *z;
{
       short  xn, zn;

       if ( form_and_or( xi, &xn, &zn ) )
       {
              if ( rest_form( xn, zn, xo, z ) )
                     return 1;
              else
                     return 0;
       }
       else
              return 0;
}
```

```
form_and_or( xi, xo, zo ) /*
--------------------------- */
short  xi, *xo, *zo;
{
        short  zi, xn;

        if ( form_1( xi, &xn, &zi ) )
        {
                if ( rest_fd( xn, zi, xo, zo ) )
                        return 1;
        }
        else
                return 0;
}

rest_fd( xi, zi, xo, zo ) /*
--------------------------- */
short  xi, zi, *xo, *zo;
{
        short  z;

        if ( theWff[xi] == '&' && form_and_or( xi+1, xo, &z ))
        {
                INIT_NODE( UND, zi, z );
                *zo = wf_ptr++;
        return 1;
    }
        else
        if ( theWff[xi] == 'v' && form_and_or( xi+1, xo, &z ))
        {
                INIT_NODE( -UND, zi, z);        /* avb = ¬(¬a&¬b) */
                WFF[zi][0] = -(WFF[zi][0]);
                WFF[z][0] = -(WFF[z][0]);
                *zo = wf_ptr++;
        return 1;
    }
        else
        {
                *xo = xi;
                *zo = zi;
        return 1;
        }
}

rest_form( xi, zi, xo, zo ) /*
----------------------------- */
short  xi, zi, *xo, *zo;
{
        short  z;

        if (( theWff[xi] == '-' ) && ( theWff[xi+1] == '>' )
                        &&  formula( xi+2, xo, &z ))
        {
                INIT_NODE( -UND, zi, z );       /* a->b = ¬(a&¬b) */
                WFF[z][0] = -(WFF[z][0]);
                *zo = wf_ptr++;
                return 1;
        }
        else
        if (( theWff[xi] == '<' ) && ( theWff[xi+1] == '-' )
                        && ( theWff[xi+2] == '>' ) && formula( xi+3, xo, &z ))
        {
                INIT_NODE( EQU, zi, z );
                *zo = wf_ptr++;
                return 1;
        }
        else
        {
                *xo = xi;
                *zo = zi;
                return 1;
        }
}

form_1( xi, xo, zo ) /*
---------------------- */
```

```
short   xi, *xo, *zo;
{
        short   f;

        if ( (unsigned char) theWff[xi] == (unsigned char) '¬' )
        {
                if ( form_1( xi+1, xo, &f ))
                {
                        WFF[f][0] = -( WFF[f][0] );
                        *zo = f;
                        return 1;
                }
                else
                        return 0;
        }
        if ( theWff[xi] == 'K' )
        {
                if (form_1( xi+1, xo, &f ))
                {
                        INIT_NODE( KNW, f, CNULL);
                        *zo = wf_ptr++;
                        return(1);
                }
                else
                        return 0;
        }
        if ( theWff[xi] == 'B' )
        {
                if (form_1( xi+1, xo, &f ))
                {
                        INIT_NODE( BEL, f, CNULL);
                        *zo = wf_ptr++;
                        return 1;
                }
                else
                        return 0;
        }
        if ( theWff[xi] == '(' )
        {
                if ( formula( xi+1, xo, &f ))
                {
                        if ( theWff[*xo] == ')' )
                        {
                                ++(*xo);
                                *zo = f;
                        return 1;
                }
                        else
                                return 0;
                }
        }
        if ( P_ATOMIC( theWff[xi]) )
        {
                WFF[wf_ptr][0] = theWff[xi];
                *zo = wf_ptr++;
                *xo = xi+1;
            return 1;
    }
        else
                return 0;
}
```

C4. File: tableau.h

```
/* ---------------------------- PROVER STUFF ------------------------ */
/*
        tableau() is the function whoe does the job. As long as there are
        branches closing and backtracking points, it will run. It stops
        (returning zero) when some branch is kept open -- i.e., no
        contradictions and no more wffs to split -- or (then returning one)
        when all branches led to contradictions and there is nothing more
        to do (no more stored branches).
```

```
*/

tableau()
{
        short           gt, val, done;

        done = val = 0;
        bckt_ptr = CNULL;

        while ( !done )
        {       /*debug(); */
                if ( CONTRADICTION )
                {
                        if ( bckt_ptr == CNULL )        /* and nothing is stored on BKCT; i.e., */
                                SUCCEED         /* no more branching...         */
                        else
                                restore_state();
                }
                else
                {
                        LOOK_FOR( UND );
                        if ( FOUND )    /* a ture conjunction, found */
                        {
                                if ( KB( OP( gt, L_SUB ) ) )
                                {
                                        T_ADD_2_MOD( LOC( gt, L_SUB ) );
                                        if ( KB( OP( gt, R_SUB ) ) )
                                        {
                                                T_ADD_2_MOD( LOC( gt, R_SUB ) );
                                                UPDATE_PRF;
                                        }
                                        else
                                        {
                                                REPL_WITH( R_SUB );
                                        }
                                }
                                else
                                if ( KB( OP( gt, R_SUB ) ) )
                                {
                                        T_ADD_2_MOD( LOC( gt, R_SUB ) );
                                        REPL_WITH( L_SUB );
                                }
                                else
                                {
                                        ADD_TRUE( gt, R_SUB );
                                        REPL_WITH( L_SUB );
                                }
                        }
                        else
                        {
                                LOOK_M_FOR( UND );
                                if ( FOUND )
                                {
                                        if ( OP_M( gt, L_SUB ) != -UND )
                                        {
                                                if ( OP_M( gt, R_SUB ) != -UND )
                                                {
                                                        T_ADD_2_MOD( LOC( gt, L_SUB ) );
                                                        TM_REPL_WITH( gt, R_SUB );
                                                }
                                                else
                                                {
                                                        ADD_MODAL( gt, R_SUB );
                                                        TM_REPL_WITH( gt, L_SUB );
                                                }
                                        }
                                        else
                                        if ( OP_M( gt, R_SUB ) != -UND )
                                        {
                                                ADD_MODAL( gt, L_SUB );
                                                TM_REPL_WITH( gt, R_SUB );
                                        }
                                        else
                                        {
                                                ADD_MODAL( gt, L_SUB );
                                                ADD_MODAL( gt, R_SUB );
```

```
    }
}
else
{
    LOOK_FOR( -UND );
    if ( FOUND )      /* a false conjunction found */
    {
        STORE_RIGHT_WFF;
        save_state( gt );

        if ( KB( OP( gt, L_SUB )) )
        {
            F_ADD_2_MOD( LOC( gt, L_SUB ) );
        }
        else
        {
            F_REPL_WITH( L_SUB );
        }
    }
    else        /* now we have modals */
    {
        LOOK_4_KB( -KNW, -BEL );
        if ( FOUND )
        {
            if( MOD[gt][0] == -KNW )
            {
                MOD[gt][0] = -U_KNW;
                if ( KB( OP_M( gt, L_SUB )) )
                {
                    F_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                }
                else
                {
                    ADD_SCOPE( ++val, gt, -NEXT( prf_ptr ) );
                    if ( !KB( OP_M(gt,L_SUB)) )
                    {
                        BLF[LOC_M(gt,L_SUB)] = 1;
                    }
                }
            }
            else  /* == -BEL */
            {
                MOD[gt][0] = -U_BEL;
                if ( KB( OP_M( gt, L_SUB )) )
                {
                    F_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                }
                else
                {
                    ADD_SCOPE( ++val, gt, -NEXT( prf_ptr ) );
                }
            }
        }
        else
        {
            LOOK_4_KB( KNW, BEL );
            if ( FOUND )      /* a true pi was found */
            {
                if( MOD[gt][0] == KNW )
                {
                    MOD[gt][0] = U_KNW;
                    if ( OP_M( gt, L_SUB ) != -UND )
                    {
                        T_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                    }
                    else
                    {
                        ADD_SCOPE( val, gt, NEXT( prf_ptr ) );
                        UPDATE_ALT;
                    }
                }
                else  /* == BEL */
                {
                    MOD[gt][0] = U_BEL;
                    if ( OP_M( gt, L_SUB ) != -UND )
                    {
```

```
                                                            T_ADD_2_MOD( LOC_M( gt, L_SUB ) );
                                                            if ( !KB( OP_M( gt, L_SUB )) )
                                                            {
                                                                  BLF[LOC_M(gt,L_SUB)] = 1;
                                                            }
                                                      }
                                                      else
                                                      {
                                                            ADD_SCOPE( val, gt, NEXT( prf_ptr ) );
                                                            UPDATE_ALT;
                                                      }
                                                }
                                          }
                                          else
                                          if ( OTHER_WORLDS )
                                          {
                                                GO_NEXT_WORLD;
                                          }
                                          else
                                                done = 1;
                                    }
                              }
                        }
                  }
            }
      }
      return 0;
}

contrad( p ) /*
----------- */
short   p;
{
        short   p2, i;

        while ( p != CNULL && !ATOMIC( PRF[p][0] ))
                --p;
        if ( p == CNULL )
                return 0;

        p2 = p-1;
        while( NON_CONTRAD )
                --p2;
        if ( p2 != CNULL)
                return 1;
        else
                return( contrad( --p ));
}

lw_contrad( m, p ) /*
------------------ */
short   p;
{
        short   p2, i;

        while ( m != CNULL && !ATOMIC( MOD[m][0] ))
                --m;
        if ( m == CNULL )
                return 0;

        p2 = m-1;
        while ( p2 != CNULL && ( MOD[p2][0] + MOD[m][0] != 0 ) )
                --p2;
        if ( p2 != CNULL)
                return 1;

        p2 = p;
        while( p2 != CNULL && ( PRF[p2][0] + MOD[m][0] != 0 || ( ( BLF[PRF[p2][1]] == 1 || PRF[p2][2]
== 0 )  && BLF[MOD[m][1]] == 1 )) )
                --p2;
        if ( p2 != CNULL)
                return 1;
        else
                return( lw_contrad( --m, p ));
}
```

```
void  save_state( r ) /*
---------------- */
short   r;
{
        short   i;

        ++bckt_ptr;
        BCKT[bckt_ptr][0] = LMK;
        BCKT[bckt_ptr][1] = PRF[r][0];
        BCKT[bckt_ptr][2] = PRF[r][1];
        BCKT[bckt_ptr][3] = PRF[r][2];
        if ( r == prf_ptr && KB( OP( r, R_SUB )) )
                --prf_ptr;      /* beta is last line on PRF */

        /* now save the modals */
        if ( mod_ptr > CNULL )
                for ( i = 0; i <= mod_ptr; ++i )
                {
                        ++bckt_ptr;
                        BCKT[bckt_ptr][0] = USD;
                        BCKT[bckt_ptr][1] = i;
                        BCKT[bckt_ptr][2] = MOD[i][0];
                        BCKT[bckt_ptr][3] = MOD[i][1];
                }

        ++bckt_ptr;
        BCKT[bckt_ptr][0] = RMK;
        BCKT[bckt_ptr][1] = prf_ptr;
        BCKT[bckt_ptr][2] = mod_ptr;
        BCKT[bckt_ptr][3] = alt_ptr;
}

void restore_state() /*
---------------- */
{
        prf_ptr = BCKT[bckt_ptr][1];
        mod_ptr = BCKT[bckt_ptr][2];
        alt_ptr = BCKT[bckt_ptr][3];
        --bckt_ptr;
        while ( BCKT[bckt_ptr][0] != LMK )
        {
                if ( BCKT[bckt_ptr][0] == USD )
                {
                        MOD[BCKT[bckt_ptr][1]][0] = BCKT[bckt_ptr][2];
                        MOD[BCKT[bckt_ptr][1]][1] = BCKT[bckt_ptr][3];
                }
                else
                {
                        PRF[BCKT[bckt_ptr][0]][0] = BCKT[bckt_ptr][1];
                        PRF[BCKT[bckt_ptr][0]][1] = BCKT[bckt_ptr][2];
                        PRF[BCKT[bckt_ptr][0]][2] = BCKT[bckt_ptr][3];
                }
                --bckt_ptr;
        }
        --bckt_ptr;
        if ( KB( BCKT[bckt_ptr][1] ) )
        {
                PRF[BCKT[bckt_ptr][0]][0] = USD;
                ++mod_ptr;
                MOD[mod_ptr][0] = BCKT[bckt_ptr][1];
                MOD[mod_ptr][1] = BCKT[bckt_ptr][2];
        }
        else
        {
                PRF[BCKT[bckt_ptr][0]][0] = BCKT[bckt_ptr][1];
                PRF[BCKT[bckt_ptr][0]][1] = BCKT[bckt_ptr][2];
        }
        if ( bckt_ptr > 0 )
        {
                        BCKT[bckt_ptr][1] = BCKT[bckt_ptr-1][1];
                        BCKT[bckt_ptr][2] = BCKT[bckt_ptr-1][2];
                        BCKT[bckt_ptr][3] = BCKT[bckt_ptr-1][3];
                        BCKT[bckt_ptr-1][0] = BCKT[bckt_ptr][0];
                        BCKT[bckt_ptr-1][1] = BCKT[bckt_ptr+1][1];
                        BCKT[bckt_ptr-1][2] = BCKT[bckt_ptr+1][2];
                        BCKT[bckt_ptr-1][3] = BCKT[bckt_ptr+1][3];
                        BCKT[bckt_ptr][0] = RMK;
        }
        else
                --bckt_ptr;
}
```

C5. File: main.c

```
/*****************************************************************
*                                                                *
*                       TTP.ZP5 -- Version 1.0                   *
*                   Cesar A. Mortari -- July 1990                *
*                                                                *
*       This program implements a tableau-like theorem           *
*       prover for the epistemic-doxastic logic ZP5.             *
*                                                                *
*                                                                *
*****************************************************************/



/* -------------------- TTP.ZP5 INCLUDES ------------------ */

#include "macros.h"
#include "prototypes.h"
#include "parser.h"
#include "tableau.h"


main()
{
        short   i, j, u;

        printf("\n*******************************************************");
        printf("\n*                 TABLEAU THEOREM PROVER FOR ZP5        *");
        printf("\n*                         Cesar A. Mortari              *");
        printf("\n*                         V1.0, July 1990               *");
        printf("\n*******************************************************");
        printf("\n\nSyntax:\n        a..t (variables), ¬, K, B, &, v, ->\n");
        printf("\nTo exit type ';'<CR>\n");
        printf("\nPlease type in a formula:\n");
        for (;;)
        {
                /* Some initializations... */

                wf_ptr = 0;
                prf_ptr = bckt_ptr = mod_ptr = alt_ptr = CNULL;

                for ( i = 0; i < STR_LEN; ++i )
                        BLF[i] = 0;

                gets( theWff );             /* input the formula to be (dis)proven */
                if ( theWff[0] == ';')              /* program ends... */
                        return 1 ;

                /* ELSE remove blanks from theWff */
                i = 0;
                j = 0;
                while ( theWff[i] != '\0' )
                {
                        if ( theWff[i] != ' ' )
                                theWff[j++] = theWff[i];
                        ++i;
                }
                theWff[j] = '\0';

                k1 = TickCount();                   /* initialize time counting */
                if ( formula( 0, &u, &i ) && theWff[u] == '\0' )
                {
                        if ( KB( WFF[i][0] ) )
                        {
                                ++mod_ptr;
                                MOD[0][0] = -WFF[i][0];
                                MOD[0][1] = i;
                        }
                        else
                        {
                                ++prf_ptr;
                                PRF[0][0] = -WFF[i][0];
                                PRF[0][1] = i;
                                PRF[0][2] = 0;
                        }
                        if ( tableau())         /* all branches were successfully closed...       */
                        {
                                k2 = (( TickCount() - k1 ) * 100) / 6;
                                putchar( 'y' );
                                putchar( 'e' );
                                putchar( 's' );
                        }
                        else            /* some open branch - tableau() returned zero      */
                        {
                                k2 = (( TickCount() - k1 ) * 100) / 6;
                                putchar( 'n' );
                                putchar( 'o' );
                        }
                        printf("\nTime: %ld ms\n\n\n", k2);
                }
                else            /* it was not a wff, so try again... */
                        printf("\nSyntax error...\n\n");
        }
}
```

D

ALG.ZP5 Listings

43rd Law of Computing:

Anything that can go wr
Error: Segmentation violation -- Core dumped.

D1. File: subf.h

```
void loop( s ) /*
-------------- */
short  s;
{
        short  p, j;

        p = a_ptr+1;
        while ( p <= p_ptr )
        {
                prf_ptr = bckt_ptr = mod_ptr = alt_ptr = CNULL;
                if ( WFF[p][3] == s && KB( WFF[p][0] ) )
                {
                        add_fi_psi( WFF[p][1] );
                        prepare_tab( p, 1 );
                        /*d_wff();*/
                        if ( tableau() )
                                WFF[p][2] = 1;
                        else
                                WFF[p][2] = 0;
                }
                ++p;
        }
}

void add_fi_psi( p ) /*
-----------------*/
short  p;
{
        short  l, r;

        if ( ATOMIC( WFF[p][0] ))
                return;
        if ( KB( WFF[p][0] ))
        {
                r = e;
                l = copy( WFF[p][1] );
                ++e;
                if ( WFF[p][2] == 1 )
                        WFF[e][0] = PST( WFF[p][0] );
                else
                        WFF[e][0] = NGT( WFF[p][0] );
                WFF[e][1] = l;
                ++e;
                WFF[e][0] = UND;
                WFF[e][1] = r;
                WFF[e][2] = e-1;
                add_fi_psi( WFF[p][1] );
        }
        else    /* conjunction */
        {
                add_fi_psi( WFF[p][1] );
                add_fi_psi( WFF[p][2] );
        }
}

short   copy( i )
short   i;
{
        short   l, r;

        if ( ATOMIC( WFF[i][0] ))
        {
                ++e;
                WFF[e][0] =  WFF[i][0];
                return e;
        }
        else
        if ( KB( WFF[i][0] ))
        {
                l = copy( WFF[i][1] );
                ++e;
                WFF[e][0] =  WFF[i][0];
                WFF[e][1] = l;
                return e;
        }
        else
        {
                l = copy( WFF[i][1] );
                r = copy( WFF[i][2] );
                ++e;
                WFF[e][0] =  WFF[i][0];
                WFF[e][1] = l;
                WFF[e][2] = r;
                return e;
        }
}

void prepare_tab( p, x ) /*
---------------------- */
short   p;
{
        short   j;

        if ( WFF[e][0] == BEL )       /* Ba is alone ; no fi psi */
        {
                ++mod_ptr;
                MOD[mod_ptr][0] = WFF[e][0];
                MOD[mod_ptr][1] = e;
        }
        else
        {
                ++prf_ptr;
                PRF[prf_ptr][0] = WFF[e][0];
                PRF[prf_ptr][1] = e;
                PRF[prf_ptr][2] = 0;
        }

        j = copy ( p );
        if ( x && WFF[j][0] < 0 )
                WFF[j][0] = -WFF[j][0];
        ++e;
        WFF[e][0] = BEL;
        WFF[e][1] = j;
        ++mod_ptr;
        MOD[mod_ptr][0] = -BEL;
        MOD[mod_ptr][1] = e;
}
```

D2 File: main.c

```
/*****************************************************************
*                                                                *
*                        ALG.ZP5 -- Version 1.0                  *
*                 Cesar A. Mortari -- August 1990                *
*                                                                *
*      This program implements the algorithm to decide           *
*      whether some wff belongs to some belief state             *
*      in the epistemic-doxastic logic ZP5.                      *
*                                                                *
*                                                                *
*****************************************************************/



/* -------------------- ALG.ZP5 INCLUDES ------------------ */

#include "macros.h"
#include "prototypes.h"
#include "parser.h"
#include "tableau.h"
#include "subf.h"


main()
{
        short  i, j, u, s;

        printf("\n**********************************************************");
        printf("\n*               BELIEF STATE ALGORITHM FOR ZP5         *");
        printf("\n*                     Cesar A. Mortari                 *");
        printf("\n*                      V1.0, August 1990               *");
        printf("\n**********************************************************");
        printf("\n\nSyntax:\n        a..t (variables), ¬, K, B, &, v, ->\n");
        printf("\nTo exit type ';'<CR>\n");
        for (;;)
        {
                /* Some initializations... */

                wf_ptr = 0;
                prf_ptr = bckt_ptr = mod_ptr = alt_ptr = a_ptr = p_ptr = CNULL;

                for ( i = 0; i < STR_LEN; ++i )
                        WFF[i][2] = WFF[i][3] = BLF[i] = 0;
                printf("\nPlease type in your 'alpha':\n");

                gets( theWff );
                if ( theWff[0] == ';')
                        return 1 ;

                /* ELSE remove blanks from theWff */
                i = 0;
                j = 0;
                while ( theWff[i] != '\0' )
                {
                        if ( theWff[i] != ' ' )
                                theWff[j++] = theWff[i];
                        ++i;
                }
                theWff[j] = '\0';

                if ( formula( 0, &u, &i ) && theWff[u] == '\0' )
                {
                        printf("\nNow type the next wff:\n");
                        gets( theWff );
                        k1 = TickCount();                   /* initialize time counting */

                        if ( formula( 0, &u, &j ) && theWff[u] == '\0' )
                        {
                                a_ptr = i;      /* add Ba to WFF */
                                e = wf_ptr;
                                WFF[e][0] = BEL;
                                WFF[e][1] = a_ptr;
                                p_ptr = j;
                                if ( WFF[p_ptr][3] > 0 )
                                        for ( s = 1; s <= WFF[p_ptr][3]; ++s )
                                                loop( s );
                                prf_ptr = bckt_ptr = mod_ptr = alt_ptr = CNULL;
                                e = p_ptr+1;
                                add_fi_psi( p_ptr );
                                prepare_tab( p_ptr, 0 );
                                for ( i = 0; i < STR_LEN; ++i )
                                        BLF[i] = 0;
                                        /*d_wff();*/
                                if ( tableau())
                                {
                                        k2 = (( TickCount() - k1 ) * 100) / 6;
                                        putchar( 'y' );
                                        putchar( 'e' );
                                        putchar( 's' );
                                }
                                else
                                {
                                        k2 = (( TickCount() - k1 ) * 100) / 6;
                                        putchar( 'n' );
                                        putchar( 'o' );
                                }
                        }
                        printf("\nTime: %ld ms\n\n\n", k2);
                }
                else                    printf("\nSyntax error...\n\n");
        }
}
```

E

References

A bibliography is usually a list of dull, dry books
placed at the end of a dull tedious book, assuring the reader
that if he reads the listed books he will be more bored still.

MURRAY LEINSTER, *The Forgotten Planet*

[AB75] ANDERSON, A.R. & BELNAP, N.D. *Entailment, the Logic of Relevance and Necessity.* Princeton, Princeton University Press, 1975.

[Al83] ALLEN, James F. "Maintaining Knowledge about Temporal Intervals". *Communications of the ACM*, 26(11): 832-843, 1983.

[Bar63] BARNES, W.H.F. "Knowing". *The Philosophical Review*, 72(1): 3-16, 1963.

[Bi65] BINKLEY, Robert. "A Theory of practical reason". *The Philosophical Review*, 74: 423-448, 1965.

[BL82] BRACHMAN, Ronald & LEVESQUE, Hector. "Competence in Knowledge representation". *Proceedings of the National Conference on Artificial Intelligence*, 1982, pp. 189-192.

[BS84] BULL, R. & SEGERBERG, K. "Basic Modal Logic". In [GG84], pp. 1-88.

[Ch80] CHELLAS, B. *Modal Logic*. Cambridge, Cambridge University Press, 1980.

[Chi63] CHISHOLM, R. "The Logic of Knowing". *The Journal of Philosophy*, 60(25): 773-795, 1963.

[Eb74] EBERLE, R.A. "A Logic of Believing, Knowing and Inferring". *Synthese* 26: 356-382, 1974,.

[FH85] FAGIN, R. & HALPERN, J.Y. "Belief, awareness, and limited reasoning". *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 491-501.

[Fi83] FITTING, Melvin. *Proof Methods for Modal and Intuitionistic Logics*. Dordrecht, D. Reidel, 1983.

[Fu88] FUHRMAN, André. *Relevant Logics, Modal Logics, & Theory Change*. Department of Philosophy and Automated Reasoning Project, Research School of Social Sciences, Australian National University, 1988.

[FV85] FAGIN, Ronald & VARDI, Moshe Y. "An internal semantics for modal logic: preliminary report". *IBM Research Report* RJ 4656 (49664), 1985.

[FV86] FAGIN, Ronald & VARDI, Moshe Y. "Knowledge and implicit knowledge in a distributed environment: Preliminary Report". In [Ha86a], pp. 187-206.

[Ga86] GALLIER, Jean H. *Logic for Computer Science*. New York, Harper & Row, 1986.

[GG84] GABBAY, D. & GUENTHNER, F. (eds.) *Handbook of Philosophical Logic, Vol. II*. Dordrecht, D. Reidel, 1984.

[Gi87] GINSBERG, Matthew L. (ed.) *Readings in Nonmonotonic Reasoning*. Los Altos, Ca., Morgan Kaufmann, 1987.

[GLB85] GEORGEFF, M., LANSKY, A., BESSIERE, P. "A procedural logic". *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 516-523.

[Go87] GOLDBLATT, Robert. *Logics of time and computation*. CSLI Lecture Notes 7. Stanford, 1987.

[Ha86a] HALPERN, J.Y. (ed.) *Proceedings of the Conference on Theoretical Aspecst of Reasoning about Knowledge* . Los Altos, Ca., Morgan Kaufmann, 1986.

[Ha86b] HALPERN, J.Y. "Reasoning about knowledge: an overview". In [Ha86a], pp. 1-17.

[Har86] HARMAN, Gilbert. *Change in View: Principles of Reasoning*. Cambridge, Mass., The MIT Press, 1986.

[HC72] HUGHES, G.E. & CRESSWELL, M.J. *An Introduction to Modal Logic*. 2nd ed. London, Methuen, 1972.

[Hck78] HAACK, Susan. *Philosophy of Logics*. Cambridge, Cambridge University Press, 1978.

[HdM89] HUDELMAIER, J. & MORTARI, C.A. "FTL: a theorem prover por classical propositional logic". Unpublished manuscript, 1989.

[Hi62] HINTIKKA, Jaakko. *Knowledge and Belief: an introduction to the logic of the two notions*. Ithaca, N.Y., Cornel University Press, 1962.

[Hi70] HINTIKKA, Jaakko. "Knowledge, Belief, and Logical Consequence". *Ajatus*, 32: 32-47, 1970.

[Hi73] HINTIKKA, J. "Surface Semantics: Definition and Its Motivation". In: LEBLANC, Hughes (ed.) *Truth, Syntax and Modality*. North-Holland, Amsterdam, 1973, pp. 128-147.

[Hi75] HINTIKKA, J. "Impossible possible worlds vindicated". *Journal of Philosophical Logic* 4: 475-84, 1975.

[Hi86] HINTIKKA, Jaakko. "Reasoning about knowledge in Philosophy: the paradigm of epistemic logic". In [Ha86a], pp. 63-80.

[HM84] HALPERN, J. & MOSES, Y. "Towards a theory of knowledge and ignorance". *IBM Research Report* RJ 4448 (48136), 1984.

[HM85] HALPERN, J. & MOSES, Y. "A guide to the modal logics of knowledge and belief: preliminary draft". *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 480-90.

[HM86] HALPERN, J. & MOSES, Y. "Knowledge and common knowledge in a distributed environment". *IBM Research Report* RJ 4421 (47909), 1984. (Revised version, 1/2/86)

[HMc84] HALPERN, Joseph & MCALLESTER, David. "Likelihood, Probability, and Knowledge". *Proceedings of the National Conference on Artificial Intelligence,* 1984, pp. 137-141.

[HS86] HALPERN, J.Y & SHOHAM, Y. "A propositional modal logic of time intervals". *IBM Research Report* RJ 5130 (53241), 1986.

[Isr] ISRAEL, David. "A Weak Logic of Knowledge and Belief: Epistemic and Doxastic Logic for the Yuppie Generation". *SRI International,* Technical Note 359.

[KK62] KNEALE, W. & KNEALE, M. *The Development of Logic.* Oxford, Clarendon Press, 1962.

[Ko82] KONOLIOE, Kurt. "Circumscriptive Ignorance". *Proceedings of the National Conference on Artificial Intelligence,* 1982, pp. 202-4.

[Ko84] KONOLIOE, K. "Belief and incompleteness". *SRI Artificial Intelligence Note 319,* SRI International, Menlo Park, 1984.

[Ko85a] KONOLIGE, K. "A computational theory of belief introspection". *Proceedings of the Ninth International Joint Conference on Artificial Intelligence,* 1985, pp. 502-508.

[Ko85b] KONOLIGE, K. "Experimental Robot Psychology". *SRI International,* Technical Note 363, 1985.

[Ko86] KONOLIGE, K. "What Awareness isn't: a sentential view of implicit and explicit belief". In [Ha86a], pp. 241-250.

[Ko87] KONOLIGE, K. "On the Relation Between Default and Autoepistemic Logic". *CSLI Report* CSLI-87-105, 1987.

[La86] LAKEMEYER, Gerhard. "Steps toward a first-order logic of explicit and implicit belief". In [Ha86a], pp. 325-340.

[Lem77] LEMMON, E.J. *The "Lemmon Notes" : An Introduction to Modal Logic.* (In collaboration with Dana Scott. Edited by Krister Segerberg.) Oxford, Basil Blackwell, 1977.

[Len78] LENZEN, W. "Recent work in epistemic logic". *Acta Philosophica Fennica* 30, 1978, pp. 1-219.

[Len80] LENZEN, W. *Glauben, Wissend und Wahrscheinlichkeit.* Wien, New York, Springer Verlag, 1980.

[Lev84] LEVESQUE, H.J. "A logic of explicit and implicit belief". *Proceedings of the National Conference on Artificial Intelligence,* 1984, pp. 198-202.

[LM84] LOPARIC,´, A. & MORTARI, C.A. "Valuations in temporal logic." *The Journal of Non-Classical Logic,* 2(1): 49-60, 1984.

[Lo77] LOPARIC,´, A. "The method of valuations in modal logic." In ARRUDA, A.I.; DA COSTA, N.C.A; CHUAQUI, R. *MATHEMATICAL LOGIC: Proceedings of the First Brazilian Conference.* New York, Marcel Dekker, 1977.

[LR86] LADNER, Richard E. & REIF, John H. "The Logic of Distributed Protocols (Preliminary Report)". In [Ha86a], pp. 207-222.

[Ma66] MAKINSON, D.C. "There are infinitely many diodorean modal functions". *Journal of Symbolic Logic,* 31(3): 406-408, 1966.

[MB83] MALIK, J. & BINFORD, T.O. "Reasoning in time and space". *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* 1983, pp. 343-345.

[Mc81] MCCARTHY, John. "Epistemological Problems of Artificial Intelligence". In: WEBBER, B.L. & NILSSON, Nils. *Readings in Artificial Intelligence.* Palo Alto, Tioga Publishing Co., 1981, pp. 459-465.

[McH69] MCCARTHY, J. & HAYES, P.J. "Some philosophical problems from the standpoint of artificial intelligence". In MELTZES, B. & MITCHIE, D. (eds.) *Machine Intelligence 4.* Edinburgh, Edinburgh University Press, 1969, pp. 463-502. (Reprinted in [Gi87], pp. 26-45)

[Mo81] MOORE, Robert C. "Reasoning about knowlege and action". In: WEBBER, B.L. & NILSSON, Nils. *Readings in Artificial Intelligence.* Palo Alto, Tioga Publishing Co., 1981, pp. 473-477.

[Mo83] MOORE, R.C. "Semantical considerations on nonmonotonic logic". *Artificial Intelligence,* 25: 75-94 1985. (Reprinted in [Gi87], pp. 127-136.)

[Mo84] MOORE, R.C. "Possible-world semantics for autoepistemic logic". *Proceedings 1984 Non-monotonic Reasoning Workshop.* New Paltz, N.Y., 1984, pp. 344-354. (Reprinted in [Gi87], pp. 137-142.)

[Mor82a] MORTARI, C.A. *Valorações para alguns sistemas de lógica do tempo.* (Valuations for some systems of temporal logical) Master dissertation, Universidade Estadual de Campinas (UNICAMP), 1982.

[Mor82b] MORTARI, C.A. "On a system of modal-temporal logic." In ALAS, O.T., DA COSTA, N.C.A., HÖNIO, C.S. *Collected Papers dedicated to Professor Edison Farah on the occasion of his retirement.* São Paulo, Instituto de Matemática e Estatística da Universidade de São Paulo, 1982, pp. 21-36.

[Ni83] NILSSON, Martin. "A logical model of knowledge". *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* 1983, p.374-376.

[Pel89] PELLETIER, F.J. "Automated Modal Logic Theorem Proving in THINKER". Unpublished manuscript, 1989.

[Pr68] PRIOR, Arthur. *Past, Present and Future.* Oxford, Clarendon Press, 1968.

[Pu70] PURTILL, Richard L. "Believing the Impossible". *Ajatus,* 32: 18-24, 1970.

[Qu80] QUINE, W.V.O. *From a Logical Point of View.* 2nd. ed., revised. Cambridge, Harvard University Press, 1980.

[Ra82] RANTALA, V. "Impossible worlds semantics and logical omniscience". *Acta Philosophica Fennica* 35, 1982, pp. 106-115.

[RB79] RESCHER, N. & BRANDOM, R. *The Logic of Inconsistency*. Rowan and Littlefield, 1979.

[Ro85] ROSENSCHEIN, Stanley J. "Formal theories of knowledge in AI and Robotics". *SRI International*, Techincal Note 362, 1985.

[RU71] RESCHER, N. & URQUHART, A. *Temporal Logic*. New York, Wien; Springer Verlag, 1971.

[Sa87] SAUNDERS, Ruth Ann. *Knowledge without belief: Fodor versus Piaget on cognitive explanations of cognitive change*. Ph.D. Thesis, University of Wisconsin–Madison, 1987.

[Tho84] THOMASON, Richmond. "Combinations of tense and modality". In [GG84], pp. 135-165.

[Va85] VARDI, M. "A model-theoretical analysis of monotonic knowledge". *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, 1985, pp. 509-512.

[Va86] VARDI, M.Y. "On epistemic logic and logical omniscience". In [Ha86a], pp. 293-305.

[XW83] XIWEN, M. & WEIDE, G. "W-JS: a modal logic of knowledge". *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, 1983, pp. 398-401.

Curriculum Vitae

I was born in 1957 in Santa Maria, Rio Grande do Sul, Brazil, where I lived until I was fifteen. In 1973 I moved to Florianópolis, capital of the state of Santa Catarina, where I began, in 1976, to study Philosophy at the Federal University of Santa Catarina (UFSC).

After concluding my studies in 1978, I moved in the following year to Campinas, São Paulo, where I began doing research in Logic for a master degree at the State University of Campinas (UNICAMP). I got my M.A in Logic and Philosophy of Science in 1983, with a dissertation called "Valuation Semantics for some systems of temporal logic".

Even during my studies there I began teaching at the UFSC, mostly Logic, sometimes Philosophy of Mathematics. Soon I got tenure at the Philosophy Department, and I remained there until the end of 1985, when I got a special leave of absence to come to Tübingen and work for a Ph.D. at the Philosophy Faculty.