Sequence Semantics for Dynamic Predicate Logic

C.F.M. Vermeulen

January 1991
Revised July 1992

Abstract

In this paper a semantics for dynamic predicate logic is developed that uses sequence valued assignments. This semantics is compared with the usual relational semantics for dynamic predicate logic: it is shown that the most important intuitions of the usual semantics are preserved. Then it is shown that the refined semantics reflects out intuitions about information growth. Some other issues in dynamic semantics are formulated and discussed in terms of the new sequence semantics.[1]

1 Introduction

In Groenendijk and Stokhof[1991a] a relational semantics for predicate logic is developed. The resulting *Dynamic Predicate Logic (DPL)* provides a synthesis of the insights of *Discourse Representation Theory* (DRT) and the elegant formalism of predicate logic. The value of such a synthesis is generally recognized, but at the same time some questions have been raised. Some people have objected that the ideas of DRT are not represented correctly in DPL. We do not intend to deal with their criticism in this paper. But also some properties of the formalism have been discovered that suggest serious problems for DPL. These problems will be the topic of this paper.

The points that we want to discuss were raised by Groenendijk and Stokhof in a paper presented at the $JELIA$-meeting [1991b]. There they compare the relational semantics of DPL with the update semantics of Veltman[1991]. Veltman provides a dynamic theory of modalities. He specifies the meaning of a modal (propositional) language in terms of operations on information states. In this approach the meaning of a formula is its potential to update information states. Groenendijk and Stokhof reformulate the semantics of DPL as an update semantics to make a comparison with Veltman's system possible. Their update formulation has the following (defining) property.

Definition 1.1 *Let ASS be the set of assignments. Let $\sigma \in \wp(ASS)$ be an information state. Let $[\![\phi]\!]_{gs} \in ASS \times ASS$ be the interpretation of ϕ as a relation on assignments. Then $(\!|\phi|\!)_{gs}$, the interpretation of ϕ as an update function, is defined by the following property:*

$$\sigma(\!|\phi|\!)_{gs} = \{g \in ASS : \exists f \in \sigma f [\![\phi]\!]_{gs} g\}$$

[1] In this paper some familiarity with the ideas and techniques of dynamic predicate logic is presupposed. We think that all that we presuppose can be found in Dynamic Predicate Logic, by Groenendijk and Stokhof[1991a]. The topics in dynamic logic that we discuss will be properly introduced, so no prior knowledge of them is required. The topics are not new however: they can be found for example in Veltman[1991], van Benthem[1989] and in Groenendijk and Stokhof[1991b].

It is this formulation of the semantics of DPL that is the basis for the comparison with Veltman's update semantics. So the properties of the semantics are discussed in terms of properties of the update functions. The following observations are made:

Proposition 1.2
- *DPL-updates are* distributive, *i.e. we always have*[2] $\sigma([\![\phi]\!])_{gs} = \bigcup\{ \{f\}([\![\phi]\!])_{gs} : f \in \sigma\}$.
- *DPL-updates are not in general* eliminative, *i.e. for some ϕ and σ* **not**: $\sigma([\![\phi]\!])_{gs} \subseteq \sigma$.

In Groenendijk and Stokhof's paper these two properties are the basis for the comparison with Veltman's system. It is pointed out that:[3]

- Veltman's updates are not in general *distributive*, i.e. for some ϕ and σ **not**: $\sigma([\![\phi]\!])_v = \bigcup\{ \{f\}([\![\phi]\!])_v : f \in \sigma\}$.
- Veltman's updates are *eliminative*, i.e. we always have $\sigma([\![\phi]\!])_v \subseteq \sigma$.

Groenendijk and Stokhof conclude that these are genuine differences between Veltman's update semantics and their own dynamic semantics, that will have to be taken into account when an attempt is made to incorporate a dynamic treatment of modalities along the lines of Veltman in DPL. But they do not seem to be worried about the fact that their semantics behaves as it does: non-eliminatively and distributively.

We have a different view on the meaning of these observations. The property of distributivity allows us to compute the result of a function on a set *pointwise*. For our update functions this means that the effects of the formula on an information state can be computed "locally". We can compute the result of an update on the whole information state by just looking at its elements. It is convenient that this holds for the operations in the DPL-semantics, but it cannot be expected to hold for all extensions. There seems to be no intuitively compelling reason why sentence meanings should be computable pointwise. The dynamic modality that Veltman considers typically is a case where one would *not* expect this. (See also section 5.2.)

For the eliminativity property things are somewhat different. In update semantics the ordering $\supseteq$ on information states corresponds to growth of information. The eliminative functions are the functions that are *monotone* along this order. Hence if an operation on information states is eliminative this means that there is an *increase* of information. And if an operation is not eliminative this means that somehow we have lost some of the information that we used to have. Therefore eliminativity is a property that we always expect if we process information, as long as typically non-monotonic features are kept out of consideration. DPL is not concerned with typically non-monotonic aspects of information processing: DPL is used for the representation of simple narrative sentences. Therefore we would not only expect eliminativity for Veltman's system, but also — and perhaps even more so — for DPL.

[2] The notation that we use here is not the one used by Groenendijk and Stokhof or Veltman, but we hope that no confusion will arise. The subscripts v and gs are used to distinguish Veltman's update functions from the DPL-updates. We will use $[\![.]\!]$-brackets for relational interpretations and $(\![.]\!)$- brackets for update interpretations throughout the paper.

[3] We do not need the definition of Veltman[1991]'s update semantics in this paper. We just give the properties of his system to illustrate the differences that Groenendijk and Stokhof[1991b] have found.

Thereby for us the non-distributivity of Veltman's system simply is a fact of life, while the fact that DPL-updates are not monotone is a problem that has to be solved.

This is so for the intuitive reason expressed above, but it is also essential for a different reason. If we accept non-monotonicity in the formal semantics, while intuitively information grows, then we seem to accept a situation in which it is not possible to talk about information in the DPL-formalism. This would be a very unfortunate situation. In fact, if DPL-semantics is not about information, then it is no longer clear what it is that DPL-semantics is about. This means that we do not know what to do with the formal system. Which properties do we have to check, what does it mean if some property does or does not hold? If we don't know what it is we are working on, then we don't know what to expect.

So we cannot afford to accept the non-monotonicity of DPL-semantics. We will propose a slight modification of the semantics (section 2), called *sequence semantics for dynamic predicate logic*. We will show that in the modified semantics the spirit of the usual relational interpretation is preserved (section 2.3). The modified semantics will give rise to an improved notion of information state (section 3) and will allow for a formulation as monotone update semantics (section 5).

The new notion of information state will be explored further (in section 4) and we will use it to shed a new light on familiar topics in dynamic semantics. First we discuss the notion of monotonicity that comes with the new notion of information state, then we attack some of the problems that the DPL-notion of inference gives rise to. Finally we will discuss a new option that becomes available in the sequence approach: down-dating (section 5.3).

It turns out that not all problems are solved simply by using sequence semantics. But the fact that some of the problems that seemed inevitable in the old formulation can be solved easily in the sequence formulation, suggests that there is hope for dynamic semantics. It encourages us to keep looking for better formalizations of the ideas of DPL until we have a system that allows us to discuss all the important issues in formal semantics properly. Then dynamic semantics will have become a serious alternative for any other formal system.

2 Sequence semantics

2.1 The language of dynamic predicate logic

In this section we present the language DPL. Usually in dynamic predicate logic, the language of (standard) predicate logic is used. We prefer to work in a slightly different language, which is very similar to the language of predicate logic but reflects the basic intuitions of dynamic semantics better. This language, DPL, is defined inductively as follows:

Definition 2.1 *Let VAR be a countable set of discourse markers, $PRED_n$ the set of n-ary predicate symbols ($n \geq 0$).*

$ATOM$, the set of atomic formulas of DPL, is the smallest set such that:

1. $\bot \in ATOM$;
2. *for each $n \geq 0$, each $P \in PRED_n$, $x_1, \ldots, x_n \in VAR$, $P(x_1, \ldots, x_n) \in ATOM$;*
3. *for each $x \in VAR$ we have $\exists x \in ATOM$.*

DPL, the language of dynamic predicate logic, is the smallest set such that:

1. $ATOM \subseteq DPL$;

2. $\phi \in DPL \ \wedge \ \psi \in DPL \ \Rightarrow \ \phi.\psi \in DPL$;

3. $\phi \in DPL \ \wedge \ \psi \in DPL \ \Rightarrow \ (\phi \rightarrow \psi) \in DPL$;

Examples of DPL-formulas are: $\exists x.\exists y,\ P(x,y,z)$, $\exists x.P(x).Q(x,x)$, $(\exists y \rightarrow P(y))$. We have conjunction (.) and implication ($\rightarrow$) as logical connectives. Negation ($\neg$) is defined as: $\neg(\phi) \equiv (\phi \rightarrow \bot)$.

As one can see, we prefer to treat the existential quantifier as an atomic formula. One reason for this is methodological: since $\exists x$ has a clear interpretation as a relation in DPL, there is no reason not to regard it as a distinct syntactic unit. We prefer to have an independent representation on the syntactic level of everything that plays an independent role in the semantics. Another reason is that this way DPL-formulas look more like $DRS's$ which makes a comparison of DPL and DRT easier. To us the language DPL seems to be a medium in which both DPL and DRT could be studied.

Universal quantification can be defined as follows: $\forall x(\phi) \equiv (\exists x \rightarrow \phi)$. The possibility of defining universal quantification in this way, is another advantage of DPL over the language of predicate logic. This definition of $\forall$ clearly reflects the intuition that we have about the meaning of $\forall$: $\forall x(\phi)$ simply means that for any x we could come up with, ϕ will hold.[4]

It should be noted that our definition of the . introduces a structural ambiguity. The formula $\phi.\psi.\chi$ can be constructed in two ways: either as the conjunction of ϕ with $\psi.\chi$ or as the conjunction of $\phi.\psi$ and χ. We will see to it that this does not cause problems for the interpretation of .-formulas (conjunctions): both "bracketings" will have the same effect in the semantics. We think that this kind of (harmless) ambiguity is also present in natural language. Also when we are considering, say, three sentences, it is not clear whether this is the result of adding the third sentence with the first two, or whether the conjunction of the last two sentences has been added to the first sentece. Therefore we feel no need to change the formal definition.

Working in this language also makes some proofs a bit easier. But it can be checked that the results that we present do not depend on the choice of DPL instead of the language of predicate logic.

2.2 The refined relational semantics

The refined version of dynamic semantics that we present is very similar to the usual relational semantics. The inductive definition of the relations that are the interpretations of the formulas of DPL is almost identical that of Groenendijk and Stokhof [1991a]. The refinement that we propose is obtained by using a richer notion of assignment. Instead of working with assignments that assign one value in the domain to each discourse referent, we use assignments that assign to each discourse marker a sequence of values. In the usual semantics we are forced to forget the current value of the variable x if we encounter the formula $\exists x$. If we consider, for example, the formula $P(x).\exists x.Q(x).\exists x.R(x)$, a pair of assignments (f,g) that is in the relational interpretation $[\![P(x).\exists x.Q(x).\exists x.R(x)]\!]_{gs}$ will only reveal a value $f(x)$ that is in the interpretation of P, and a value $g(x)$ that is in the interpretation of R, but we have no way of remembering a value in the domain that is in the interpretation of Q.

[4] This is also the intuition that is behind the proof rule for $\forall I$ in natural deduction for example.

We think that this is where the usual semantics goes wrong. If we throw away values of some variable, then we lose track of the restrictions on the values of this variable. These restrictions on the value of a variable are the way in which information gets represented in DPL-semantics. And once we have thrown away this information, there is no way of recovering it.

If we work with assignments that have sequences as values, we can see to it that the value that is in Q is remembered. For example, if we find a pair (k, h) in the refined relational interpretation $[\![P(x).\exists x.Q(x).\exists x.R(x)]\!]$, such that $k(v) = \langle \ldots, p\rangle$ and $h(v) = \langle \ldots, p, q, r\rangle$: this tells us that $p \in I(P)$, $q \in I(Q)$ and $r \in I(R)$. This way no information is lost. We will call these assignments that have sequences as values, sequence valued assignments.

Definition 2.2 *1. $SASS$, the set of sequence valued assignments, is the set that consists of all functions $f : VAR \to DOM^*$. (Here DOM is the domain of interpretation, and $DOM^* = \bigcup\{DOM^n : n \geq 0\}$.)*

2. *For a sequence $\langle d_1, \ldots, d_n\rangle \in DOM^*$ we call $end(\langle d_1, \ldots, d_n\rangle) = d_n$.*

3. *For a sequence $\langle d_1, \ldots, d_n\rangle \in DOM^*$ we call $pd(\langle d_1, \ldots, d_n\rangle) = \langle d_1, \ldots, d_{n-1}\rangle$.*

4. *For two sequences $\langle d_1, \ldots, d_n\rangle$, $\langle e_1, \ldots, e_m\rangle \in DOM^*$, we define their concatenation: $\langle d_1, \ldots, d_n\rangle * \langle e_1, \ldots, e_m\rangle = \langle d_1, \ldots, d_n, e_1, \ldots, e_m\rangle$.*

It should be noted that the last component of the sequence $f(x)$, $end(f(x))$, is the *current* value of x. We also use the notation $pd(f(x))$, the predecessor of $f(x)$, for the sequence that is left if we remove $end(f(x))$. So we have $f(x) = pd(f(x)) * \langle end(f(x))\rangle$. If the empty sequence is assigned to x ($f(x) = \langle\rangle$), then we call $f(x)$ undefined and we say $end(f(x)) = \langle\rangle$. The fact that we allow the empty sequence as a value introduces some form of partiality into our semantics. Therefore we sometimes call the elements of $SASS$ *partial*, sequence valued assignments.

It is also possible to use partial assignments in the usual relational semantics. Instead of working with total assignments, Groenendijk and Stokhof could have used partial assignments $f : VAR \hookrightarrow DOM$. (Here the partiality is indicated by the $\hookrightarrow$.) Let's call the set of partial assignments $PASS$.[5]

In such an approach new questions concerning partiality have to be answered. For example, if we want to define the truth of a formula, ϕ say, in such an approach, we have two options:

> ϕ is true iff for any partial assignment f, there is a partial assignment g such that...;

or

> ϕ is true iff for any partial assignment f *defined on the free variables of* ϕ, there is a partial assignment g... .

If we always choose for the second kind of answer, this gives us a semantics virtually equivalent to the usual relational semantics. We do not give the details of this approach: it requires but a straightforward adaptation of the usual semantics. We will use the notation

[5] Note that $ASS \subseteq PASS \subseteq SASS$.

$[\![\phi]\!]_{pgs}$ to refer to this version of the relational semantics.[6] Following the first option will give us something quite different. It might be interesting to see what happens in that kind of semantics, but we will not do that here.

Note that the use of partial assignments instead of total assignments is a step in the right direction: if we use partial assignments, we only have to consider the variables that actually occur in the formula. This avoids confusion about the role that the values of other variables play. We can see about which variables the formula actually gives information.

We can also extract this kind of information from the interpretation with total assignments. For example, if in the interpretation of a formula ϕ the value of x is restricted, then it is clear that x occurs in ϕ. And also if in the interpretation of ϕ the value of x can change, we can conclude that x occurs in ϕ. But the total assignments do not reveal *all* the information about the occurrence of variables.

For example, if $\phi = (P(x) \rightarrow P(x))$, then ϕ is a statement about x. We admit that the information that ϕ gives about x is trivial, but still ϕ is a statement about x and not about any other variable. In the interpretation with total assignments we cannot see this: $[\![\phi]\!]_{gs}$ simply contains all pairs (f, f). But if we use partial assignments then this information becomes available in the semantics: if $(f, f) \in [\![\phi]\!]_{pgs}$, then f will have to be defined on x, but not necessarily on any other variable. So now we can see from the interpretation of ϕ that x occurs in ϕ. In other words, if we use partial assignments the interpretation can tell us about which variables formulas make a statement.[7]

But clearly the use of partial assignments is not enough to get a good representation of all the information a formula can contain: we do not only want to make a difference between the variables that occur and the variables that do not occur, but about the variables that occur, we also want to know in how many different roles they occur.[8]

Now before we give the refined version of the relational semantics, where this information is present, we introduce an important technical notion: we define what it means for one (sequence valued) assignment to be a x-variant of another (sequence valued) assignment.

Definition 2.3 *Let $x \in VAR$, $V \in VAR^*$, $f, g \in SASS$ be given.*

1. *We say that g is a x-variant of f,*

$$f\langle\!\langle x \rangle\!\rangle g \text{ iff } \exists d \in DOM : g(x) = f(x) * \langle d \rangle \wedge \forall y : (y \neq x \rightarrow f(y) = g(y)).$$

2. *We define the notion V-variant for sequences of variables V as follows:*

$$f\langle\!\langle\ \langle x \rangle\ \rangle\!\rangle g \Leftrightarrow f\langle\!\langle x \rangle\!\rangle g;\ f\langle\!\langle\ V * \langle x \rangle\ \rangle\!\rangle g \Leftrightarrow \exists h \in SASS : f\langle\!\langle\ V\ \rangle\!\rangle h \wedge h\langle\!\langle x \rangle\!\rangle g.$$

The relation $\langle\!\langle x \rangle\!\rangle$ allows us to choose a new value for x. We do not throw away the old value: we build up a sequence of values instead. Note that the x-variant relation is *not* symmetric. In fact we have $f\langle\!\langle x \rangle\!\rangle g \ \rightarrow\ \neg\ g\langle\!\langle x \rangle\!\rangle f$. Another important property of these relations is that $\langle\!\langle V \rangle\!\rangle$ is the same as $\langle\!\langle V' \rangle\!\rangle$ whenever V' is a permutation of V. So we have, for example, $\langle\!\langle\ \langle x, y, x \rangle\ \rangle\!\rangle = \langle\!\langle\ \langle y, x, x \rangle\ \rangle\!\rangle$.

Now we will define the refined relational semantics.

[6] See for example Kamp and Reyle[1990], Dekker[1992] or Fernando[1991] for a presentation of semantics with partial assignments. Kamp and Reyle [1990] use partial assignments for the semantics of DRT.

[7] Here the analogy between formulas and computer programs might be helpful: even if you want to run a trivial program, you have to make sure all variables that actually occur are properly declared.

[8] In Dekker[1992] we see that even in the presence of partial functions strong syntactic restrictions are required to obtain eliminativity. In Dekker's approach all formulas in which a variable occurs in more than one role (such as $\exists x.P(x).\exists x.Q(x)$) are regarded as meaningless.

Definition 2.4 *1. For each $\phi \in DPL$ we define $[\![\phi]\!] \subseteq SASS \times SASS$, the meaning of ϕ as a relation on sequence valued assignments. Let f, g, h, k be sequence valued assignments, I an interpretation function for the predicates of DPL.*

$$f[\![P(x_1, \ldots, x_n)]\!]g \Leftrightarrow f = g \wedge (end(f(x_1), \ldots, end(f(x_n))) \in I(P)$$
$$f[\![\exists x]\!]g \Leftrightarrow f\langle\!\langle x \rangle\!\rangle g$$
$$f[\![\phi.\psi]\!] \Leftrightarrow \exists h\ f[\![\phi]\!]h[\![\psi]\!]g$$
$$f[\![(\phi \rightarrow \psi)]\!]g \Leftrightarrow f = g \wedge \forall h : f[\![\phi]\!]h \Rightarrow \exists k : h[\![\psi]\!]k$$

2. *We define valid inference as follows: let $\phi, \phi_1, \ldots, \phi_n, \psi \in DPL$. Then:*

$$\phi \models_{[\![}\ \psi \Leftrightarrow \forall f, g : f[\![\phi]\!]g \Rightarrow \exists h : g[\![\psi]\!]h$$
and
$$\phi_1, \ldots, \phi_n \models_{[\![}\ \psi \Leftrightarrow \phi_1. \ \ldots\ .\phi_n \models_{[\![}\ \psi.$$

The definition is very similar to the original definition by Groenendijk and Stokhof. The main difference is the kind of assignments that is used. This has some interesting consequences. Note for example that in the definition of $[\![\phi.\psi]\!]$ the h such that $f[\![\phi]\!]h[\![\psi]\!]g$ is uniquely determined by f, g.

Now if we look at $[\![\phi]\!]$, we can find almost all the information that is revealed by ϕ: if we look carefully at the variables on which pairs (f, g) that are in $[\![\phi]\!]$ are defined, then we can find out which variables are free in ϕ, which variables occur quantified and how often a variable is quantified over. For example, if we consider $[\![P(x).\exists x.Q(x)]\!]$, we will see that for all pairs (f, g) that are in this relation, we have that both f and g are defined on x (i.e. $f(x) \neq \langle\rangle$),which means that x must occur in the formula; the fact that f will always be defined on x, means that there is a *free* x in the formula. We will see that the length of the sequence $g(x)$ is at least two, which means that x occurs in the formula in two (possibly different) roles. Since x already occurs freely, we can infer from this that there must be exactly one occurrence of $\exists x$ in the formula. We will also see that for any other variable y, there is a pair (f, g) where $f(y) = g(y) = \langle\rangle$, indicating that y does not occur in our formula.

It is also possible to make this information explicit in the relational semantics. If we add two components to our indices, we can build up the set of the free variables of a formula and the sequence of the quantified variables quite easily. Then $[\![.]\!]$ would become a relation on triples (A, S, f) where A contains the free variables and S gives the information about the quantified variables and $f \in SASS$. We will not make this precise here, but the reader can see how it could be done in the static semantics that we will present later.

2.3 A comparison

In this section we compare the refined relational semantics with the usual relational semantics. We will show that for any ϕ the partial relational interpretation, $[\![\phi]\!]_{pgs}$, can be constructed out of the relational interpretation $[\![\phi]\!]$. This way we also establish a relation between $[\![\phi]\!]$ and $[\![\phi]\!]_{gs}$, since $[\![\phi]\!]_{gs}$ is just the restriction of $[\![\phi]\!]_{pgs}$ to total assignments.

We establish the relation between $[\![\phi]\!]$ and $[\![\phi]\!]_{pgs}$ with a mapping Φ that is defined as follows:

Definition 2.5 *We define $\Phi : \wp(SASS \times SASS) \rightarrow \wp(PASS \times PASS)$ as follows: $\Phi(R) = \{(g, f) : \exists (k, h) \in R : \forall x \in VAR : f(x) = end(h(x)) \wedge g(x) = end(k(x))\}$.*

As one can see, we construct a relation between partial functions out of a relation between sequence valued functions by restricting our attention to the current values of the sequence valued functions. The result of such a restriction is in general not a total function since we allow the value $\langle\rangle$ for sequence valued functions.[9]

Now our claim about Φ is the following:

Proposition 2.6 *For all $\phi \in DPL$ we have $\Phi([\![\phi]\!]) = [\![\phi]\!]_{pgs}$.*

The proof of proposition 2.6 is an induction on the complexity of ϕ. It can be found in the appendix.

The relation that Φ establishes does *not* enable us to construct $[\![\phi]\!]$ from $[\![\phi]\!]_{pgs}$ for all ϕ: it is a homomorphism rather than a isomorphism. As an example of this we consider the formula $\phi = \exists x.P(x).\exists x.Q(x)$. If we know $[\![\phi]\!]_{pgs}$, we only can see which values in the domain are such that they have property $I(Q)$ (by checking which values of x can occur as $f(x)$ in a pair $(g, f) \in [\![\phi]\!]_{pgs}$). If we want to have $[\![\phi]\!]$, we also have to know which part of the domain has property $I(P)$.

It is possible, however, to define the relation $[\![.]\!]$ in terms of the relation $[\![.]\!]_{pgs}$. In other words: if we know $[\![\phi]\!]_{pgs}$ for *all* ϕ, then we can give $[\![\phi]\!]$ for all ϕ. As an example again consider $\exists x.P(x).\exists x.Q(x)$. Knowing $[\![\exists x.P(x).\exists x.Q(x)]\!]_{gs}$ does not suffice to find $[\![\exists x.P(x).\exists x.Q(x)]\!]$. But if we know *both* $[\![\exists x.P(x)]\!]_{gs}$ and $[\![\exists x.Q(x)]\!]_{gs}$ then we do have enough information to construct $[\![\exists x.P(x).\exists x.Q(x)]\!]$. In general, if we know how ϕ decomposes into conjuncts, then we can construct the sequence valued interpretation of ϕ from the sequence valued interpretation of the conjunctions. This weaker correspondence suffices to guarantee that the notions of validity for $[\![.]\!]$, $[\![.]\!]_{pgs}$ and $[\![.]\!]_{gs}$ are equivalent. Therefore we have but one notion of validity, in spite of the subtle differences between the different semantics for DPL. First we give the definitions of valid inference for $[\![.]\!]_{gs}$ and $[\![.]\!]_{pgs}$.

Definition 2.7 *For ϕ, $\psi \in DPL$ we define:*

1. $\phi \models_{gs} \psi \Leftrightarrow \forall f, g \in ASS : f[\![\phi]\!]_{gs} g \rightarrow \exists h \in ASS : g[\![\psi]\!]_{gs} h$;
2. $\phi \models_{pgs} \psi \Leftrightarrow \forall f, g \in PASS :$

   $f[\![\phi]\!]_{pgs} g$ *and g is defined on the free variables of* $\psi \rightarrow \exists h \in PASS : g[\![\psi]\!]_{pgs} h$.

All definitions of $\phi \models_x \psi$ are based on the same idea: if we are able to make a ϕ-step, then we are always able to make a ψ-step. Now our claim is that also formally all notions of valid inference coincide, i.e. we claim:

Corollary 2.8 *For any ϕ, ψ the following clauses are equivalent:*

1. $\phi \models_{[\!]} \psi$;
2. $\phi \models_{pgs} \psi$;
3. $\phi \models_{gs} \psi$.

[9] In this context we mean by $f(x) = end(\langle\rangle)$ that $f(x)$ is undefined.

The proof can be found in the appendix.

In the proof $2 \Rightarrow 1$ we need more than just $[\![\psi]\!]_{pgs}$to be able to construct the required sequence valued assignment h: we needed all the $[\![\psi_i]\!]_{pgs}$ (where the ψ_i are the formulas of which ψ is the conjunction). This confirms our remark above about the relation between $[\![\psi]\!]$ and $[\![\psi]\!]_{pgs}$ not being an isomorphism.

By now we think that we have given sufficient proof of the fact that in our refinement of the relational semantics the spirit of the Groenendijk and Stokhof semantics is preserved: we can reconstruct the original relations from our refined relational semantics and we have preserved the original notion of entailment.

3 Static semantics

3.1 Information structures

In this section we will introduce the notion of an information structure. Then we will interpret DPL-formulas as information structures and this will be our static semantics of DPL. We will show that this interpretation is compatible with the relational interpretation of the preceding section: we will be able to construct the relational interpretation from the static interpretation and we will see that the information structure that we associate with a formula is just the set of possible outputs of the relation $[\![.]\!]$.

At first sight the fact that dynamic DPL allows for a static semantics may seem odd. Does this mean that dynamic semantics is static after all?[10] It might be helpful to think about the situation in terms of *representation* or *coding*. The information structures that we will define can be used to *represent* (or *code*) the relations that we have defined above. Not all relations on $SASS$ will have such a representation, but it turns out that all DPL-interpretations can be represented by some information structure. We already have a similar situation in the original formulation of the DPL-semantics. There conditions, such as $P(x)$, can be represented by a set, the set of assignments that satisfy the condition. Given this set the relational interpretation of the condition can be constructed. For example, $P(x)$ can be represented by the set $\mathbf{P}= \{f \in ASS : f(x) \in I(P)\}$. Then we can define $[\![P(x)]\!]_{gs}= \{(f,g) : f = g \wedge g \in \mathbf{P}\}$. But for other formulas such a representation as a set is not available.

With the new definition of the relational semantics and the new notion of information structure a similar reconstruction will be possible for all DPL-interpretations. But still the dynamic perspective is useful. First because it is the relational formulation that has led to our definition of information structure in the first place. And second because there are still interesting notions in dynamic semantics that cannot be represented statically, even by the new information structures. Here one can think, for example, of Veltman's dynamic modalities (also see section 5.2).

What is the information revealed by a formula? The answer to this question will motivate our notion of information structure. Surely, a formula gives information about the discourse markers that occur in the formula. We represent this by considering all the values that such a discourse marker can take, i.e. our information structures will contain assignments of values to the discourse markers that occur in the formula. The range of values of a discourse marker is restricted by the conditions that we find in the formula. So we will not find all assignments in the information structure, but only those that satisfy the restrictions that are expressed by the formula.

[10] I would like to thank an anonymous referee for pointing out this source of confusion.

Some of these discourse markers are introduced explicitly in the formula: they occur in the scope of a quantifier. For these variables a value gets *set* whenever we interpret the formula. But a formula also might say something about discourse markers that it does not introduce itself: a formula might contain free variables. It would be unreasonable to treat these free variables on a par with the quantified variables. The formula does not *instruct* us to assign a value to them, it simply assumes that we have done this already: a formula *asks* for a value on its free variables. The information about the free variables that the formula gives specifies further which value we should have assigned to them. We include this information into our information structures by restricting the assignments according to these conditions. To make sure that we do not lose sight of the different status that the restrictions on free variables have, we include in our structures components that tell us which variables occur freely and which variables are quantified over in the formula.

If the set of assignments that we have in our information structures really has to give some information about the discourse markers of the formula, it has to have, in some sense, this set of discourse markers as its "domain". It is not yet clearly defined what it means for an assignment to have that property. To make the notion precise we give the following definitions.

Definition 3.1 *1. With each sequence $S \in VAR^*$ we associate a partial function $\alpha_S : N \rightarrow VAR$ as follows:*

$\alpha_{\langle x \rangle}(0) = x$;
$\alpha_{\langle x \rangle}(n+1) =$ *undefined for* $n \in N$;
$\alpha_{\langle x \rangle * S}(0) = x$;
$\alpha_{\langle x \rangle * S}(n+1) = \alpha_S(n)$ *for* $n \in N$.

2. *We define the length of a sequence S as follows:*

$$length(S) = max\{n \in N : \alpha_S(n) \text{ is } defined\} + 1.$$

3. *For each sequence $S \in VAR^*$ we define what it means for a sequence valued assignment $f \in SASS$ to be defined on S.*

f is defined on $\langle x \rangle$ iff $length(f(x)) \geq 1$;
f is defined on $S * \langle x \rangle$ iff f is defined on S and
$length(f(x)) > min\{length(g(x)) : g \text{ is } defined \text{ on } S\}$.

4. *For a set $V \subseteq VAR$ we write $seq(V)$ for the sequence made out of the elements of V. (Here we use some standard enumeration of VAR to order the elements of V in $seq(V)$. Such an enumeration is available since VAR is countable.)*

5. *We define the relation $\leq_l$ on assignments in $SASS$:*

$$\text{Let } f, g \in SASS. \text{ We say } f \leq_l g \text{ iff } \forall x : \exists \sigma_x : \sigma_x * f(x) = g(x).$$

6. *If $f \leq_l g$, then we say that f is a restriction of g.*

In definition 3.1.1 we associate a function defined on an initial segment of N with each sequence. The function α_S gives for $n \in N$ the nth element of the sequence S. The

relation of such a function to the sequence is the same as the relation of a characteristic function to a set. They are interchangeable and we only distinguish them notationally to avoid confusion.[11] Note how α_S can be used to give a concise definition of the length of a sequence (3.1.2).

The other clauses are concerned with making clear how we check whether a sequence S is in the domain of a sequence valued function f. Definition 3.1.5 makes precise a notion of the *extension* of a sequence valued assignment. If $f \leq_l g$, then $g(x)$ may be longer than $f(x)$, but the extra values of $g(x)$ are added in front. We also call such an extension an irrelevant or *left* extension. If g extends f to the left, we have for each x that $f(x)$ and $g(x)$ agree on the relevant values, i.e. the last few values.[12]

As an example, consider a model where $I(P) = \{a\}$ and an assignment f with $f(x) = \langle a \rangle$. Then f satisfies the condition $P(x)$: the current value of x is in $I(P)$. If we extend f to the left to g such that $g(x) = \langle b, c, d, e, b, d, a \rangle$, then for the formula $P(x)$ this extension of the value of x is irrelevant. g is as good as f even though the new values on x are not in $I(P)$. The current value of x still is a, so g also satisfies the condition. We can cut of an irrelevant part of $g(x)$ to get for example $g'(x) = \langle b, d, a \rangle$. Usually we will only be interested in the values of a sequence valued assignment on the variables that actually occur in some formula. For example, in the case of $P(x)$ we are only interested in the current value of x, in the case of $P(x).\exists x.Q(x)$ we want to know the last two values of x, etc.

This leads to the following definition of information structures:

Definition 3.2 *$INFO \subseteq \wp(VAR) \times VAR^* \times \wp(SASS)$ such that $(A, S, F) \in INFO$ iff*

1. *$f \in F \Rightarrow$ f is defined on $seq(A) * S$;*
2. *$f \in F \wedge f \leq_l g \Rightarrow g \in F$;*
3. *$f \in F \wedge g \leq_l f \wedge g$ is defined on $seq(A) * S \Rightarrow g \in F$.*

(Here A is for asking, *S is for* setting *and F is for* function.*)*

In an information structure (A, S, F) we want the assignments in F to contain information about A and S. This means that all the $f \in F$ have to be defined on both A and S: $seq(A) * S$ is the minimal domain for the $f \in F$ (condition 1). It also means that the values outside this minimal domain should not matter. Hence the $f \in F$ should at least be defined on A and S, but any extension outside this minimal domain should be allowed. Furthermore the values outside the minimal domain should not be restricted: value restrictions code information and we only want information about A and S, not about other variables. Conditions 2 and 3 have this effect: given an $f \in F$ defined on the minimal domain, 2 says that *all* its (left-)extensions also are in F and 3 says that if $g \in F$, then also all restrictions of g are in F, as long as they are defined on $seq(A) * S$.

The sequence S will give the exact order of all the quantifiers in a formula. These are the variables for which the formula *sets* a value. Since we will only be interested in the number of times a quantifier $\exists x$ occurs in a formula — and not in the question whether $\exists x$ occurs before $\exists y$ or after — we do not really need a sequence as a second component: a multiset gives all the information we want. But using multisets would complicate things technically:

[11] Sometimes finite sequences are simply defined to *be* functions defined on initial segments of N.

[12] In the appendix we give a lemma that says exactly what we mean when we say that left extensions are irrelevant.

if we work at the same time with sets (in the first component), multisets (in the second component) and sequences (from the third component), we would have to introduce even more auxiliary definitions and technical tricks. Already the use of both sets and sequences has forced us to introduce more auxiliary techniques than we would like: the notion of the range of a sequence ($range(\alpha_S)$) and a function seq to make sequences out of sets.

So we allow ourselves some extra structure in the second component, but this is for convenience only.[13]

The set A will contain the variables that occur freely in the formula. A formula cannot by itself *set* these variables to the right value: it has to take over the value that the context provides. One could say that the formula — and therefore also the information structure — *asks* the context for a value for such a variable.

Given an information structure (A, S, F) it is possible to reconstruct $seq(A) * S$ from F:[14] we just have to find the minimal domain of the $f \in F$. Therefore it is possible, given F and S to find A and given F and A to find S.[15] This suggests that we only need two and not three components in an information structure. But that is not the case: it is impossible to define the domain conditions on the F in an information structure and distinguish the status of the A-like and the S-like part of the minimal domain without actually mentioning all three components. It *is* true that — once the definition is given — the last two of the three components suffice to find the first.

Our notion of information structure is motivated solely by considering formulas in DPL. Therefore it might seem rather artificial. To give some more motivation we want to point out the similarity to discourse representation structures ($DRSs$). A DRS also consists of a component of discourse markers and a set of assignments that is to express the restrictions on these discourse markers that we find in discourse. The set A that we have in our information structures can be compared to the set of *anchored* discourse markers.[16] In DRT these anchors are mainly used for deictic expressions. This is also something that we can use the A-variables for.

For further motivation for the presence of this third component in our information structures we compare our situation with the practice of computer programming. In a program we are not allowed to work with undefined variable names. Nevertheless names can occur in a program that are not declared in the program itself. We call these names constants, but this does not mean that they never have to be defined. Instead of being defined in the program, a constant can be said to be defined in the program environment. Such a situation can also occur in discourse theory, where we may want to study a discourse fragment and not the whole of discourse. We then assume some *discourse environment* in which the free variables are defined. Free variables could stand for proper names, for example. We simply assume that proper names have been introduced properly. We would not want to be forced to introduce them in every bit of discourse in which we want to use them; they are declared in the discourse environment.

It is also possible that we are simply not *able* to introduce the free variables properly. Such a situation arises, for example, if one overhears part of a conversation. Suppose there is a discussion going on between two people about a third person. To the two conversationalists it probably is clear who this third person is. Therefore he (or she) will probably not be

[13] We do not mean that it is really difficult to use multisets, but we feel it would only confuse the issue.

[14] Up to permutation!

[15] Again, up to permutation.

[16] An anonymous referee pointed out this analogy of the A-variables and anchors.

represented as a free variable in their representation of the discourse. But if we miss the introduction of this discourse marker, overhearing only a part of the conversation, we will be forced to represent this person as a free variable. We will assign all the properties that are assigned to this person to our free variable. But our understanding of the discourse is incomplete until we find out who it was that they were talking about; we only have a partial understanding of that piece of discourse, because our discourse environment is not rich enough.

3.2 Static interpretation

Now we will define the interpretation of DPL-formulas as information structures. We will assign an element of $INFO$ to each formula ϕ, that we will call (A_ϕ, S_ϕ, F_ϕ).

Definition 3.3 *1. For each DPL-formula ϕ, we define the static interpretation of ϕ, $[\phi]$, inductively as follows:*

$$[P(x)] = (\{x\}, \langle\rangle, \{f : end(f(x)) \in I(P)\})$$
$$[\exists x] = (\emptyset, \langle x\rangle, \{f : \exists g : g\langle\langle x\rangle\rangle f\})$$
$$[\phi.\psi] = (A_\phi \cup (A_\psi \backslash range(S_\phi)), S_\phi * S_\psi, \{f \in F_\psi : \exists g \in F_\phi : g\langle\langle S_\psi\rangle\rangle f\})$$
$$[(\phi \rightarrow \psi)] = (A_\phi \cup (A_\psi \backslash range(S_\phi)), \langle\rangle, \{f : \forall g \in F_\phi : f\langle\langle S_\phi\rangle\rangle g \rightarrow \exists h \in F_\psi g\langle\langle S_\psi\rangle\rangle h\})$$

2. We define valid inference as follows: $\phi \models_{[} \psi \Leftrightarrow \forall g \in F_\phi \exists h \in F_\psi : g\langle\langle S_\psi\rangle\rangle h$

The first component of $[\phi]$ is the set of the free variables in ϕ.[17] In the second component of $[\phi]$ we build up the sequence of the quantified variables. That allows us to keep track of the order in which variables are (re)introduced.[18]

Example:

1. $[P(z).\exists x] = (\{z\}, \langle x\rangle, \ldots)$;
2. $[Q(x,y).\exists x.R(y,x)] = (\{x,y\}, \langle x\rangle, \ldots)$;
3. $[P(z).\exists x.Q(x,y).\exists x.R(y,x)] = (\{z\} \cup (\{x,y\} \backslash \{x\}), \langle x\rangle * \langle x\rangle, \ldots) = (\{z,y\}, \langle x,x\rangle, \ldots)$.

Here 3 is the conjuction of 1 and 2. We see that the x that is free in 2, is not free in the conjunction; it is bound by the $\exists x$ in 1. Therefore is has to be removed from the set of free variables. Note that a variable can be free only once, while it can occur many times in the sequence of the quantified variables.

The assignments in the third component will all be defined on the free and the quantified variables. It is also easy to check that the third components of the interpretations satisfy the other conditions in the definition of $INFO$. This means that $[\phi] \in INFO$ for any $\phi \in DPL$.

There is an obvious correspondence between this static semantics and the refined relational semantics of the preceding section:

[17] We have compared the variables in A with anchors, but while free variables can get bound in larger contexts, anchors will always remain anchors. So it would be better to consider the A-variables as *temporary* anchors.

[18] Note that here $[\exists x.P(x).\exists y.P(y)] \neq [\exists y.P(y).\exists x.P(x)]$. If we use multisets these two information structures become equal.

Proposition 3.4 *Let* $[\phi] = (A_\phi, S_\phi, F_\phi)$. *Then:*

1. $f[\![\phi]\!]g \;\Leftrightarrow\; g \in F_\phi \wedge f\langle\!\langle S_\phi \rangle\!\rangle g$.
2. $\phi \models_{[} \psi \;\Leftrightarrow \phi \models_{[\![} \psi$.

(The proof is omitted.) If we had used a version of $[\![.]\!]$ that has three components (cf. p.7), this would mean that: $[\phi] = range([\![\phi]\!])$ (or $\uparrow[\![\phi]\!]$ in the notation of Groenendijk and Stokhof[1991b]). The proposition has the following corollary:

Corollary 3.5 *For any* ϕ, ψ *the following clauses are equivalent:*

1. $\phi \models_{[\![} \psi$;
2. $\phi \models_{[} \psi$;
3. $\phi \models_{pgs} \psi$;
4. $\phi \models_{gs} \psi$;

Proposition 3.4 and corollary 3.5 guarantee that our interpretation of formulas in $INFO$ preserves the spirit of dynamic semantics. This means that a relational formulation is not essential for the dynamic semantics of DPL. This may seem surprising at first, but maybe not if we recall that already in DRT we have a kind of static semantics that covers almost the same evidence as DPL.

Also recall our discussion in 3.1 where it is pointed out that for certain extensions of dynamic semantics a relational formulation will be essential.[19]

4 Topics in dynamic semantics

4.1 The general point

Now that we have an improved notion of information, we want to discuss some topics in dynamic semantics in terms of it. One of the issues we will discuss is monotonicity. Remember that we said (section 2.2) that probably the defect of the original formulation of DPL was that we were sometimes forced to forget the value of a variable. If we consider $[\![P(x).\exists x.Q(x).\exists x.R(x)]\!]_{gs}$, we will only find the p such that $p \in I(P)$ and the $r \in I(R)$, but no $q \in I(Q)$. This means that if there is any $q \in I(Q)$, we would have $[\![P(x).\exists x.R(x)]\!]_{gs}$=$[\![P(x).\exists x.Q(x).\exists x.R(x)]\!]_{gs}$. This makes it impossible to have an intuitively acceptable notion of information based on these relations. Indeed, it makes it impossible to answer questions about information by looking at the semantics.

In our set up it is possible to define an ordering of the semantic objects, information structures, that corresponds to our basic intuitions about the informativity of the formulas. We will define this ordering and show how it works. This is the ordering along which we want to check the monotonicity of our semantics.

Then we will try to use our ordering for the study of dynamic inference. This will cause some problems that suggest that our ordering is not "the right one", at least not for all purposes.

[19] Also see section 5.

4.2 Ordering information structures

We will introduce an ordering of information structures, that will be motivated by the intuition that *more discourse contains more information.* We will discuss the basic properties of this ordering. There are basically two ways in which we can give more information in discourse. First we can say more about the objects that we already were talking about. We would then add restrictions on discourse markers, and as a consequence eliminate some assignments. The other way in which we can add information, is by introducing a new object in discourse.

It is difficult to find a piece of discourse that does just this. One could think of the indefinite article 'a',[20] but larger discourse fragments that only *introduce* and do not *restrict* are hard to find. Usually as soon as we introduce some object we say something about it as well. But maybe we can illustrate the two kinds of information be contrasting the following two sentences.

Example:

1. There is a unicorn.

2. There is a man.

The first sentence clearly is informative in both senses. An object is introduced for us to talk about and a very interesting claim about the object is made. From this sentence we can infer that unicorns exist. Probably this is what the speaker wants to say with this sentence.

From the second sentence we can infer that men exist. But probably this will not be what the speaker is trying to tell us in this sentence. Here this claim is merely a side effect. The main goal of the speaker probably is to introduce an object that he wants to talk about. The first claim about it is already made — that it is a man — but, surely, more will follow. Such introductory acts are essential in the chain of information exchange.

In other words, although both sentences are informative in both ways, the second sentence *mainly* serves to introduce the object for further discussion. The first sentence also makes a remarkable claim about the object it introduces.[21]

Other, more artificial examples can be found in in mathematics. In proving a theorem, in arithmetic for example, a phrase such as

Let n be given.

is often used. Clearly the purpose of this phrase is just to make it possible to talk about some number.

In DPL-syntax these two ways of giving information correspond, roughly, to two kinds of formulas: the first kind of information is typically represented by DPL-conditions, and the second kind goes together with the DPL-quantifier. We should be careful with this correspondence: when we find a formula $\exists x$ in DPL, we are not sure that this x will really stand for a new object, it might just be another name for an old object.[22] Nevertheless, it is the formula $\exists x$ that makes it possible to talk about new objects and therefore it seems

[20] Even for 'a' the situation is not so easy. Sometimes 'a' does more then just introduce an object. Think, for example, of generic uses of the indefinite article.

[21] I want to thank the referees for helping me out with this example.

[22] Also see section 4.4.

reasonable to say that $\exists x$ contains positive information (in this sense). In our semantics the two ways of giving information can be represented as follows:

Definition 4.1 *We define an ordering* $\geq$ *on* $INFO$. *Let* (A,S,F), $(A',S',F') \in INFO$ *be given. Then* $(A,S,F) \geq (A',S',F')$ *iff*

1. $S' = S * S''$ *for some* S'';
2. $A' \subseteq A$;
3. $\forall f' \in F' : \exists f \in F : f \langle\!\langle S'' \rangle\!\rangle f'$.

We have defined $\geq$ in such a way that the smaller information structures are more informative. In this definition 1 represents (part of) the second idea about increase of information: the more informative state defines all the discourse markers that are defined in the larger state and maybe some more.

If we have a situation in which $S'' = \langle\rangle$, the third clause reads as $F' \subseteq F$. This is the easiest way to see that the first idea is also reflected by the definition: if no more discourse markers are introduced, giving more information simply means eliminating some assignments. What 3 actually says is that every assignment in F' should be an extension (to the right, so *not* irrelevant!) of an assignment in F. This corresponds to a more refined notion of the elimination of assignments: we do not want to say that an assignment is eliminated if it has "grown". An assignment is not eliminated if it gets a larger domain: it can only be said to be eliminated if it does not reoccur at all, not even with a larger domain. If we want to consider assignments as possibilities, it is this refined notion of elimination that corresponds to the idea of eliminating a possibility. The sequence valued assignments are like possible histories of information. Just as history will not have to be revised, because of developments in the future, assignments are not eliminated if they become "longer". Now we can see that 3 corresponds to this notion of elimination of possibilities: every $f' \in F'$ is an extension of an $f \in F$, but some assignments $f \in F$ might not reoccur in F', they are eliminated. Note that 2 usually follows from 1 and 3; the only exception is $F = F' = \emptyset$. (We have $(A,S,\emptyset) \in INFO$ for all A,S.) It would be interesting to understand what this exception means. It seems to say that when a contradiction arises, i.e. $F = \emptyset$, we have to make a choice: either we say that all contradictions contain the same information or some give different information from others. Here we have chosen the last option, not for any intuitive reason but because it makes the formalism easier to handle.

Before we show what this way of ordering information structures means for our DPL-interpretations, we give some abstract properties of the ordering.

Proposition 4.2 $\geq$ *is a partial order, i.e.*

1. $(A,S,F) \geq (A',S',F') \ \wedge \ (A',S',F') \geq (A'',S'',F'') \ \Rightarrow \ (A,S,F) \geq (A'',S'',F'')$;
2. $(A,S,F) \geq (A',S',F') \ \wedge \ (A',S',F') \geq (A,S,F) \ \Leftrightarrow \ (A',S',F') = (A,S,F)$.

Proof:

2 '$\Leftarrow$' is obvious. '$\Rightarrow$': Clearly the antecedent implies $S = S'$ and $A = A'$. Therefore we find both $F \subseteq F'$ and $F' \subseteq F$, i.e. $F = F'$.

1 $S'' = S' * S^*$ for some S^*, and $S' = S * S^\circ$ for some S°. So $S'' = S * S^\circ * S^*$. Also, if $f'' \in F''$, we have $f' \in F'$ such that $f' \langle\!\langle S^* \rangle\!\rangle f''$, and for this f' we have $f \in F$ such that $f \langle\!\langle S^\circ \rangle\!\rangle F'$, and therefore $f \langle\!\langle S^\circ * S^* \rangle\!\rangle f''$.$\Box$

4.3 Monotonicity

In this section we show that our ordering of information structures works fine for the easiest way of giving more information in discourse: we show that larger discourse fragments are more (or: not less) informative in our sequence based interpretation. We call this property of our semantics monotonicity. As was explained in the introduction it is our view that any good semantics for ordinary, narrative discourse should have this property. It is simply true that we do not lose information if we continue our story. Of course, this does not mean that we *never* forget information, but forgetting is not the result of ordinary narrative discourse. And it is this kind of discourse that we want to represent in DPL. So we want the following property for our semantics:

Proposition 4.3 *Let $\phi, \psi \in DPL$ be given. Then we have $[\phi] \geq [\phi.\psi]$.*

Proof: $[\phi.\psi] = (A_\phi \cup (A_\psi \backslash range(S_\phi)), S_\phi * S_\psi, \{f \in F_\psi : \exists g \in F_\phi : g \langle\!\langle S_\psi \rangle\!\rangle f\})$, $[\phi] = (A_\phi, S_\phi, F_\phi)$. Now the proposition holds by definition of $\geq$.□

At this point we can see in which way our semantics is an improvement of the usual relational semantics for DPL. By using sequence valued assignments, we have enriched the semantics in a natural way. As a result we get the possibility of a systematic and abstract discussion of information in DPL-semantics. We now have semantic objects that are rich enough to make such an approach possible. The definition of information structures and our ordering of these structures are examples of these new possibilities. They have enabled us to see that DPL-semantics is monotone.[23] It is not claimed here that these two notions are all we will ever need, but they show what kinds of things are possible in this richer environment, using sequence valued assignments.

4.4 Inference

4.4.1 Dynamic inference in general

Inference is a notoriously difficult topic in dynamic semantics. Different branches of dynamic semantics have given rise to different notions of inference and some have even produced more than one. The DPL-notion of inference is defined as follows:

$$\phi \models \psi \Leftrightarrow \forall f, g : f [\![\phi]\!]_{gs} g \rightarrow \exists h : g [\![\psi]\!]_{gs} h.$$

We have seen that this notion of inference is preserved in all the reformulations of DPL-semantics that we have considered.

Update semantics has produced its own notions of inference. One of them reads as follows[24]:

$$\phi \models_v \psi \Leftrightarrow \forall \sigma : \sigma([\![\phi]\!]_v) \subseteq \sigma([\![\phi.\psi]\!]_v)$$

Both notions of inference make sense in the context in which they arise. So it seems that dynamic semantics in general does not have *one* notion of inference. Instead we find a whole spread of inference relations. Of course, this gives rise to the question what the common

[23] We will get back to this point in section 5.1.

[24] Notation as in introduction: v stands for Veltman.

features of these relations are. Or — to put it ironically — is there any relation that is not a dynamic inference relation?

This turns out to be a surprisingly difficult question.[25] But here we do not aim at solving this general problem. Instead we restrict ourselves to the DPL-notion of inference.[26] We are especially interested in the way in which the inference relation fits into our algebra of information structures. We know that for all sorts of formal systems there is a nice fit between the algebraic semantics and the inference relation. For propositional logic $\models$ coincides with the ordering in Boolean (or, in the intuitionistic case, Heyting) algebra. For predicate logic and modal (propositional) logic it is the inclusion relation on satisfaction sets. For linear logic the situation is less straightforward: here $\models$ can be defined in terms of $\oplus$.[27] Now what kind of relation holds between the DPL-inference relation and the ordering that we have defined on $INFO$?

4.4.2 Inference and information

The idea that there might be a connection between the inference relation of DPL and the ordering on $INFO$, is not just inspired by the fact that this is so for other formal systems. There also is a clear intuition that inference and information are related concepts. It seems that if we can *infer* ψ from ϕ, this must be because ϕ gives all the *information* that we need to conclude that ψ. And, conversely, if ψ contains no more *information* than ψ, then, surely, we should be able to *infer* ψ from ϕ.

In this paper we have been concerned with the information contained in DPL-formulas from a different perspective. We have been trying to model the idea that "more discourse contains more information". This has given rise to a notion of information structure. Of course it is our hope that this notion of information will also allow us to say something about inference.

There are different relations between $\models$ and $\leq$ that we could discuss. We will start with the most obvious one and we will consider some other options as we proceed. The first guess is that the inference relation simply *is* the $\leq$-relation on information structures:

$$\phi \models \psi \;\Leftrightarrow\; [\phi] \leq [\psi].$$

One example which supports this is:

$$\exists x.P(x) \models P(x).$$

But unfortunately the first counterexample is not far away. If we turn around the $\models$-sign in the above example we get a case where $\phi \models \psi$ but *not* $[\phi] \leq [\psi]$.

A somewhat weaker relation, which is still possible in view of these examples, is:

$$\phi \models \psi \;\Leftarrow\; [\phi] \leq [\psi].$$

For, it is not the case that $[P(x)] \leq [\exists x.P(x)]$, so the counterexample that we had, is not a counterexample for this weaker correspondence. This weaker correspondence would suggest

[25] More about this issue can be found in Van Benthem[1991].

[26] Note that the fact that this notion of inference has survived the reformulations in this paper, is evidence that it is indeed at the core of DPL, even if in the more general picture it is just one of a number of candidates.

[27] A similar situation exists in Pratt's action logic (Pratt[1991]).

that our ordering on $INFO$ is too strong: if $[\phi] \leq [\psi]$, then $\phi \models \psi$, but even if *not* $[\phi] \leq [\psi]$, $\phi \models \psi$ can still hold.

But this is not the case: we can have ϕ and ψ such that $[\phi] \leq [\psi]$ but *not* $\phi \models \psi$, as the following example shows.

$$[P(x).\exists x.\neg(P(x))] \leq [P(x)], \text{ but not: } P(x).\exists x.\neg(P(x)) \models P(x).$$

That the first relation holds is true in view of the monotonicity result that we established in the previous section: more discourse contains more information. But it is also clear that *not* $P(x).\exists x.\neg(P(x)) \models P(x)$, since this would imply that the same instance of x would both have property P and not have property P.

It is clear what goes wrong in this example: in $\phi \models \psi$ the binding between the antecedent ϕ and the consequent ψ makes that we are talking about the same x both having and not having property P. But the x in the antecedent that has property P is not the same as the x that has property *not* P.

We could try to prevent this kind of anaphoric confusion as follows. Instead of comparing $[\phi]$ and $[\psi]$, we compare $[\phi]$ and $[\phi.\psi]$. If we do this, we can take into account the bindings between ϕ and ψ already if we are looking at $\leq$. So this should help to prevent the unpleasant surprises that these bindings, that are essential for the $\models$-relation, cause in cases as the above. Therefore our next guess is that:

$$\phi \models \psi \Leftrightarrow [\phi] \leq [\phi.\psi].$$

Note that this guess reflects the same intuition about the relation between information and inference: if $\phi \models \psi$, then what we learn from $\phi.\psi$ is no more than what we learn from ϕ.[28] In other words: given ϕ, ψ contains no new information. Also note that for most of the systems that we mentioned above the two guesses coincide formally. For example, for propositional logic and Boolean algebras we have that for two propositions P_1 and P_2, $[P_1] \leq [P_2]$ iff $[P_1 \wedge P_2] \leq [P_1]$. It might be the case that in our situation this formulation with the conjunction is simply more suitable, since in DPL anaphoric bindings are so important.

In the new situation the counterexample that we had no longer works. For now $[\phi.\psi] = [P(x).\exists x.\neg(P(x)).P(x)]$ has to be compared with $[\phi] = [P(x).\exists x.\neg(P(x))]$. We see that not $[\phi] \leq [\psi]$. Therefore we would not expect $\phi \models \psi$ in the first place.

But again there is a counterexample. The counterexample is embarrassingly simple — and in fact it also defeats our earlier guesses — but also very instructive.

$$\exists x.P(x) \models \exists y.P(y).$$[29]

Here we see that *not* $[\phi] \leq [\phi.\psi]$, simply because $\phi.\psi$ says something about more variables than ϕ alone. Since the introduction of new variables counts as an increase of information according to $\leq$, we do not find $[\exists x.P(x)\exists y.P(y)] \geq [\exists x.P(x)]$.

Here we see the main problem for the comparison of $\models$ and $\leq$. $\leq$ is based on the idea that as a rule new variables are introduced to give new information. But the variables that are introduced in the consequent of $\models$ typically are *not* introduced for this purpose. They are there to make claims about old information.[30] If we say $\exists x.P(x) \models \exists y.P(y)$, than we

[28] Also note the similarity with Veltman's notion of entailment.

[29] Note that also $\exists x.P(x) \models \exists x.P(x)$ is a counterexample. So the choice of the variable in the consequent is not important.

[30] The same holds for the variables in the consequent of an implication.

do not mean that some unknown object y has the property P, but we claim that when we know $\exists x.P(x)$, we already know an object with property P.

It seems that we are at a dead end: in order to model the idea that more discourse contains more information, we had to count the introduction of variables as informative acts. This was an important motivation for our definition of $\leq$. We have seen that in the context of inference we typically consider discourse that is not supposed to contain new information, but nevertheless can contain new variables. These are conflicting requirements on the ordering of information.

4.4.3 Multi-dimensional information algebras

The conflict that we have seen seems irrepairable: we cannot think of a way in which it could be solved. We think that in DPL information is given by conditions on variables. These conditions can be represented as restrictions of the values of the variables. Any sensible notion of information state for DPL should show the variables that the information is about as well as the restrictions on these variables that embody the information. So the problem that we have sketched will arise for any sensible notion of information structure that one might come up with. Always the same question will arise: do more variables mean more information or not? And always the answer will be both yes and no.

Therefore our conclusion must be that there is more than one way to look at the information of a DPL-formula. These different ways give rise to different orderings of the information structures. We have already seen two perspectives on information that give rise to two different orderings of information structure. One is the perspective where we consider ϕ and $\phi.\psi$ and ask ourselves which of the two we would prefer to hear. Of course we would choose $\phi.\psi$, since *it contains more information* than ϕ alone.

In the other perspective we imagine that we are in a situation that we have heared ϕ and we wonder whether we still want to hear ψ. Of course we only want hear ψ, if given ϕ *it contains new information.* This situation, in which we consider ϕ and ψ in a specific order and not as unordered alternatives, gives rise to a different ordering of information structures. We call this the *diachronic* information order. This is the ordering that should correspond directly to $\models$. The situation where we can choose between ϕ now or $\phi.\psi$ now gives rise to a *synchronic* ordering of information structures. This is the situation that we have considered in section 4.2.[31]

We can illustrate the difference between these two ways of looking at information with an example from DPL. Consider the formula $\phi \equiv \neg P(x).\exists x.P(x)$. This is an example of a DPL-formula that does not entail itself. Therefore it will behave funnily in a diachronic information ordering. But in a synchronic information ordering it will not behave funnily: synchronically each formula is of course as informative as itself.[32]

We can think of these different orderings as the dimensions of the information algebra. So the conclusion is that we have to work in a multi-dimensional information algebra. Of course this cannot be the final word about information orderings. Just as with the study of dynamic inference the fact that there are several sensible information orderings gives rise to further questions. We would like to know what kind of relations count as information orderings, i.e. how many dimensions there are in our information algebra. Is there any relation on information structures that is not an information ordering?

[31] The terminology is taken from Visser[1991], who applies the distinction in a slightly different context.

[32] This example is due to one of the referees.

At the moment it seems to us that there is a feature that all ordering relations should have in common. Let's assume that our information structures contain both a set of variables[33] — the variables that the information is about — and a set of functions embodying the restrictions on the values of the variables. This is not only true for our information structures but also — as we have argued — for any reasonable alternative. Comparing the information contained in these information structures always amounts to comparing the restrictions on variables that we find in the different information structures. If for every variable in one structure, σ say, we can find a variable in the other structure, σ', that is at least as severely restricted, then we are inclined to say that σ' contains *more* information than σ.

This could be tested with a mapping from the variables in σ' to the variables in σ. If we can find a mapping such that the restrictions on the images of the variables are at least as severe as the restrictions in σ, then we would say that σ' is more informative than σ. The severity of the restrictions can, of course, be compared by looking at the assignments. We intend to develop this general idea about the ordering of information states elsewhere. Here we just check how the two information orderings that we have seen relate to this general idea.

We find that both the synchronic and the diachronic information ordering embody this idea. But both notions have some extra conditions on the variables that we are allowed to compare, conditions on the mapping from σ' to σ as it were. If we test whether $(A, S, F) \leq (A', S', F')$, we check whether for any $f' \in F'$ there is an $f \in F$ that has the same values for the variables in S' as f itself. Here this comparison of variables in different information structures is effectuated by the condition $f\langle\langle S''\rangle\rangle f'$ (where S'' is such that $S = S' * S''$). This condition tells us which variable occurrences have to be compared. For example, if $(A, S, F) = (\emptyset, \langle x, x\rangle, F)$ and $(A', S', F') = (\emptyset, \langle x\rangle, F')$, then we will compare the last value of $f'(x)$, not with the last value of $f(x)$, but with the value before last: $f(x) = f'(x) * \langle d\rangle$ for some d.

For $\models$ the relation with our general information ordering is less straightforward. The relation between entailment and the general ordering can be made precise, but the technical details would take us beyond the scope of this paper. Suffice it to make the connection intuitively clear.

In checking intuitively that something like $\exists x.P(x).\exists y.Q(y).R(y) \models \exists z.Q(z)$ holds, we try to find (for z) in the antecedent a variable that satisfies at least the condition Q. Here this variable is y. There are no restrictions on the variable that we can choose: any variable will do as long as it satisfies Q.

If we have a free variable in the conclusion (as in $\exists x.P(x) \models P(x)$), then the situation is different. This time we do not have a free choice at all. A free variable in the conclusion can be bound by a variable in the antecedent. If this is the case, then this is the variable we should choose. So in the example we can only compare the x in the conclusion with the x in the antecedent. If the variable is not bound by the antecedent (as in $\models (P(x) \rightarrow P(x))$), then the conclusion has to hold for all values of x. So we can see that also for $\models$ we have to compare variables, taking into account some restrictions.

The conclusion of this section is that there is no straightforward algebraic relation between $\leq$ and $\models$. In this respect DPL-semantics is less well behaved then the formal systems we mentioned above. But the orderings on information structures that $\leq$ and $\models$ lead to,

[33] Strictly speaking we should say "a set of variable instances", for we will have to distinguish different occurrences of variables (just as we do in sequence semantics), according to the different roles one variable can play.

seem to be instances of one general scheme for the comparison of information. This general information order will get more attention elsewhere.

5 Update semantics

5.1 Update semantics

In the preceding sections we have given interpretation of DPL in terms of assignments that have sequences as values. We have checked that our semantics is faithful to the original DPL-semantics (in section 2.3) and we have seen that we can give both a relational and a static formulation of our semantics. In this section we formulate sequence semantics as update semantics. We will see that such a formulation is available. Then we will discuss the issue of eliminativity again. We have addressed this issue already in terms of the static semantics (section 3.3), but if we formulate the notion of monotonicity in update style, this will make the comparison with the original discussion by Groenendijk and Stokhof[1991b] more straightforward.

First we define an operation on information states, that we call the *merger*.

Definition 5.1 *We define the merger of information structures,* $\bullet : INFO \to INFO$, *as follows:*

$$(A, S, F) \bullet (A', S', F') \;=\; (A \cup (A' \backslash (range(\alpha_S)), S * S', \{f \in F' : \exists g \in F : g \langle\!\langle S' \rangle\!\rangle f\}.$$

We use this operation to define the interpretations of formulas as update functions on information structures.

Definition 5.2 *For a DPL-formula* ϕ *we define the update function* $(\!|\phi|\!) : INFO \to INFO$ *as follows:*[34]

$$(A, S, F)(\!|\phi|\!) \;=\; (A, S, F) \bullet [\phi].$$

Here we have defined $(\!|\phi|\!)$ in terms of $[\phi]$, but it is an easy exercise to show that we can also define $(\!|\phi|\!)$ directly in such a way that the defining property holds.

Note that the update functions make it possible to built up the static interpretation: for example, if $(A, S, F) = [\psi]$, then $(A, S, F,)(\!|\phi|\!) = [\phi.\psi]$.[35] Hence we can find the interpretation of a conjunction by updating the state of no information, $(\emptyset, \langle\rangle, SASS)$, step by step. In other words:

Proposition 5.3 *Let* $\phi_0, \ldots, \phi_n \in DPL$ *be given. Then we have:*

$$[\phi_0. \;\ldots\; .\phi_n] \;=\; (\ldots((\emptyset, \langle\rangle, SASS)(\!|\phi_0|\!))\ldots)(\!|\phi_n|\!).$$

We can also give a definition of inference in terms of update functions.

Definition 5.4 *Let* $(\emptyset, \langle\rangle, SASS)(\!|\phi|\!) \;=\; (A, S, F)$ *and* $(\emptyset, \langle\rangle, SASS)(\!|\psi|\!) \;=\; (A', S', F')$. *Then we define:*

$$\phi \models_{up} \psi \;\Leftrightarrow\; \forall f \in F : \exists g \in F' : f \langle\!\langle S' \rangle\!\rangle g.$$

[34] We use postfix notation for update functions.

[35] This explains the notation for the merger as $\bullet$: it is the semantic analogue of '.' .

Because of the close relationship with the static interpretation that we have established in proposition 5.3, it can be checked easily that this notion of inference coincides with the one(s) discussed before.

Corollary 5.5 $\phi \models_{up} \psi \Leftrightarrow \phi \models \psi$.

Now we can formulate the monotonicity property in terms of update functions.

Proposition 5.6 *Let $\phi \in DPL$ be given. Then $(\!|\phi|\!)$ is monotone decreasing, i.e.:*

$(A,S,F) \geq (A,S,F)(\!|\phi|\!)$ *for all* $(A,S,F) \in INFO$.

The proof of the proposition again relies on the correspondence between the static and the update interpretation. Because of this correspondence the result simply follows from the monotonicity result for the static interpretation (section 4.3).

We see that for the improved notion of information state the DPL-updates are monotone, or — in the terminology of Groenendijk and Stokhof[1991b] — eliminative. Now that we have discovered this improved notion of monotonicity, we can check what this property amounts to in terms of the relational semantics. We find, by a careful reconstruction, the following reformulation of monotonicity:

Proposition 5.7 *Let $\sigma \in ASS$, $\phi, \psi \in DPL$ be given. Then:*[36]

Monotonicity $\forall f \in \sigma(\!|\phi.\psi|\!)_{gs} \exists g \in \sigma(\!|\phi|\!)_{gs}$ *such that* $g \langle\!\langle range(\alpha_{S_\psi}) \rangle\!\rangle f$.

Truth property $\sigma(\!|\phi.\psi|\!)_{gs} \neq \emptyset \Rightarrow \sigma(\!|\phi|\!)_{gs} \neq \emptyset$.
Hence the truth of $\phi.\psi$ implies the truth of ϕ.

Proof: [Monotonicity] From the monotonicity property for our update semantics we learn that $\sigma \bullet [\phi] \bullet [\psi] \leq \sigma \bullet [\phi]$. For $\sigma = (\emptyset, \langle\rangle, SASS)$, this means that all f' in the static interpretation of $\phi.\psi$ are extensions of some g' in the static interpretation of ϕ. Because of the relation between the sequence semantics and the original relational semantics we see that this means that for any $f \in range([\![\phi.\psi]\!]_{gs})$ there is a $g \in range([\![\phi]\!]_{gs})$ that differs from f only on the variables occurring in ψ. This proves the proposition for $\sigma = (\emptyset, \langle\rangle, SASS)$. The proof of the general case is completely analogous.

[Truth property] Follows immediately from the Monotonicity property.□

Of course, this is not anything like the eliminativity property that Groenendijk and Stokhof considered. Since they were not careful in the choice of their notion of information state, they defined an inappropriate notion of eliminativity, one that did not correpond to the notion of information growth. Since for Veltman's system this is exactly what the eliminativity property is about, the resulting comparison of DPL and Veltman's update semantics was confused. Now we are in a position to clear up the confusion and we find the monotonicity property that we would expect for DPL.

[36] Here we abuse the notation $\langle\!\langle . \rangle\!\rangle$: we use it for a set instead of a sequence. Of course we mean the usual notion of resetting a function here, where $f\langle\!\langle X \rangle\!\rangle g$ allows us to reset all the values of the variables in the set X.

5.2 Might

Veltman does not introduce update semantics just as a nice way to present dynamic semantics. He has some substantial applications in mind (see Veltman[1990]) in the dynamic semantics of modalities. The simplest system that Veltman applies the techniques of update semantics to is propositional logic with an operator $\Diamond$, *might*. The update semantics enables us to give a dynamic interpretation to *might*. What we mean by this is shown by the following example:

Example:

1. It might be raining. ... It is not raining.

2. It is not raining. ... It might be raining.

The first sentence says that we first think that it might be raining and later find out that it is not raining. This is all right. But in 2 we still think that it might rain after we have found out that it is not raining. That is nonsense. This example shows that a dynamic treatment of *might* is needed. Only a dynamic *might* could explain why the first sentence seems acceptable, and the second not. Veltman has succeeded in giving an elegant semantics for propositional logic with $\Diamond$ that deals with this phenomenon.

In this section we will see whether an easy extension of our system with a dynamic modality is available. We will show that it is possible to define a dynamic might operator in our system. The semantics of this operator, $\Diamond$, cannot be given "pointwise". We mean that it is not possible to compute the effect of $\Diamond\phi$ in some complex state, by first computing its effect on the atomic substates and then simply adding the results to find the effect on the complex state. This is in contrast with what we have seen so far: for our DPL-updates we have:

Proposition 5.8 *Update functions for DPL-formulas are "pointwise":*
$(A,S,F)([\![\phi]\!]) = (A',S',\bigcup\{G_f : f \in F\})$,
where for all $f \in F$ $(A,S,\{f\})([\![\phi]\!]) = (A',S',G_f)$.

The fact that such a result cannot be obtained for the semantics of $\Diamond$ is not a defect of our semantics: it is an essential property of the meaning of $\Diamond$. $\Diamond\phi$ induces a test on our current state of information: the test succeeds if ϕ is compatible with our information. Then it leaves the state of information unchanged. If ϕ is incompatible with what we already know, the test fails. Then the result $\Diamond\phi$ is total confusion: the information of $\Diamond\phi$ gives a contradiction.

If we try to perform such a test bit by bit, we will (possibly) throw away some information, since it is incompatible with ϕ, while we leave other bits of information intact. Then, if we add up the resulting bits of information, we could only retrieve some of the information that we started with. This is in contradiction with the test character of $\Diamond\phi$. Hence a pointwise approach to the semantics of *might* does not stand a chance.

We can define the concept of acceptability or compatibility that is associated with $\Diamond$, as follows:

Definition 5.9 *ϕ is acceptable in (A,S,F) iff* $\exists f \in F : \exists g \in F_\phi : f\langle\langle S_\phi \rangle\rangle g$.

Remember that we defined :

ϕ is valid in (A,S,F) iff $\forall f \in F : \exists g \in F_\phi : f\langle\langle S_\phi \rangle\rangle g$.

So, while validity says that ϕ should hold in all possible cases, acceptability says that ϕ should hold in at least one possible case. In this context we can think of the set F as the set of possibilities (or possible information histories).

Now we can explain the meaning of $\Diamond\phi$ as follows: checking whether $\Diamond\phi$ holds in a situation (A,S,F) means checking whether ϕ is acceptable in (A,S,F). So we define:

Definition 5.10 *For each ϕ we define $(\!|\Diamond\phi|\!)$ as follows:*[37]

$(A,S,F)(\!|\Diamond\phi|\!) = (A,S,F)$ *if ϕ is acceptable in* (A,S,F);

$(A,S,F)(\!|\Diamond\phi|\!) = (A,S,\emptyset)$ *else.*

It can happen that for some formula ϕ, $(\Diamond\phi).\neg(\phi)$ is acceptable, while $\neg(\phi).(\Diamond\phi)$ is not acceptable, just as in the example above. Consider, for example, the formula $P(x)$. We find that, in a model where there is some $p \in I(P)$, while not $I(P) = DOM$:

$$(\emptyset, \langle x \rangle, \{f \in SASS : \mathit{length}(f(x)) \geq 1\})(\!|\Diamond P(x)|\!)(\!|\neg P(x)|\!) =$$

$$(\emptyset, \langle x \rangle, \{f \in SASS : \mathit{length}(f(x)) \geq 1\})(\!|\neg P(x)|\!) =$$

$$(\emptyset, \langle x \rangle, \{f \in SASS : \mathit{length}(f(x)) \geq 1 \wedge \mathit{not} : \mathit{end}(f(x)) \in I(P)\}),$$

but also:

$$(\emptyset, \langle x \rangle, \{f \in SASS : \mathit{length}(f(x)) \geq 1\})(\!|\neg(P(x))|\!)(\!|\Diamond P(x)|\!) =$$

$$(\emptyset, \langle x \rangle, \{f \in SASS : \mathit{length}(f(x)) \geq 1 \wedge \mathit{not} : \mathit{end}(f(x)) \in I(P)\})(\!|\Diamond P(x)|\!) =,$$

$$(\emptyset, \langle x \rangle, \emptyset).$$

This confirms that what we have defined is a *dynamic* might operator.

However, the non-commutativity of $(\!|\phi|\!)$, which is the clue to its dynamic character, holds only for a restricted class of formulas. We find the non-commutativity only if ϕ contains free variables. This can be understood as follows:

Suppose that ϕ contains no free variables, i.e. $A_\phi = \emptyset$. We know that

ϕ is acceptable in (A,S,F) iff $\exists f \in F\ \exists g \in F_\phi : f\langle\langle S_\phi \rangle\rangle g$.

But if $A_\phi = \emptyset$ then this is the case exactly if:

$\forall f \in F\ \exists g \in F_\phi : f\langle\langle S_\phi \rangle\rangle g$.

(This follows from the irrelevance lemma that we proof in the appendix. The remark there about free variables is important here.) But this is the definition of validity. So formulas without free variables are acceptable iff they are valid.

And if ϕ is valid, then $\neg(\phi)$ is not acceptable, so $(A,S,F)(\!|\neg(\phi)|\!) = (A,S,\emptyset)$. But then we find that in case ϕ does not contain free variables

[37] Note that $(\!|\Diamond\phi|\!)$ is not representable as an information structure!

ϕ is acceptable in $(A,S,F) \Leftrightarrow$

$(A,S,F)(\!|\Diamond\phi|\!) \;=\; (A,S,F) \Rightarrow$

$(A,S,F)(\!|\Diamond\phi|\!)(\!|\neg(\phi)\,|\!) \;=\; (A,S,F)(\!|\neg(\phi)|\!) \;=\; (A,S,\emptyset).$

Hence:

$$(\!|\Diamond\phi.\neg(\phi)|\!) = (\!|\neg(\phi).\Diamond\phi|\!).$$

This restricts the applicability of our $\Diamond$ as a dynamic might operator. We can understand the restriction from the technical point of view, but it does not seem to make sense intuitively. If something *might* exist, then usually it does not follow that it *does* exist. Still the fact remains that $\Diamond$ gives us a consistency test for DPL as an operation on information structures. And for a non-trivial fragment of the language this consistency test has a dynamic character.

5.3 Down-dating

In the section on monotonicity we explained why the semantics of DPL should be monotone: DPL is to be the language of ordinary discourse in which more and more information is revealed by the speaker and gathered by the hearer. We also said that for some other situations it might be handy to have a language and a semantics of forgetting or down-dating, as we will call it. In this section we extend DPL with atomic formulas $x\mathsf{E}$ — read as "x exit" — for any variable x, that will be interpreted as an instruction to forget the current value of x. We will see that down-dating helps us to formulate old ideas more elegantly.

The interpretation of $x\mathsf{E}$ is essentially relational: $x\mathsf{E}$ does not give information, it is purely an action.[38] We define the relational interpretation of $x\mathsf{E}$ as follows:

Definition 5.11 $[\![x\mathsf{E}]\!] = \{(f,g) : pd(f(x)) = g(x) \wedge (y \neq x \rightarrow g(y) = f(y))\}$. (*again* $pd(\langle\rangle) = \langle\rangle$).[39]

In this extension of the language we can have local variables: for example in $\exists x.\phi.x\mathsf{E}$. We can also give an update formulation of the meaning of a down-date.

Definition 5.12 $(A,S,F)(\!|x\mathsf{E}|\!) \;=\; (A',S',\{g : \exists f \in F : g\langle\!\langle x \rangle\!\rangle f\})$,

where S' is obtained from S by removing the last x. If there is no occurrence of x in S, $S' = S$ and we remove x from A: $A' = A\backslash\{x\}$. If x does not occur in A either: $A' = A$.

Note that $(\!|x\mathsf{E}|\!)$ works best if there is at least one x available in (A,S,F). If there is no x, then $(A,S,F)(\!|x\mathsf{E}|\!) = (A,S,F)$.

Now it is possible to establish another relation between $[\![\phi]\!]$ and $[\![\phi]\!]_{gs}$. As one can see: $[\![\exists x]\!]_{pgs}$=$[\![x\mathsf{E}.\exists x]\!]\cap PASS \times PASS$ and $[\![\exists x]\!]_{gs}$=$[\![x\mathsf{E}.\exists x]\!]\cap ASS \times ASS$. This is no surprise: the original interpretation of $\exists x$ told us to *replace* a value of x. In the refined semantics this can be established by two seperate actions: first throw away the old value of x with $x\mathsf{E}$, then add the new value with $\exists x$. Thereby we are able to translate ordinary dynamic predicate logic into the enriched language by replacing all quantifiers $\exists x$ by $x\mathsf{E}.\exists x$. If we call this translation $\circ$, we find:

Proposition 5.13 *Let $f,g \in PASS$. Then: $f[\![\phi^\circ]\!]g \Leftrightarrow f[\![\phi]\!]_{pgs}g$.*

[38] This means that it can not be represented by some information structure.

[39] pd is defined in definition 2.2.3.

Corollary 5.14 *Let $f, g \in ASS$. Then: $f[\![\phi]\!]_{gs} g \Leftrightarrow f[\![\phi^\circ]\!]g$.*

(The proof is omitted.) The restriction that $f, g \in PASS$ is added because $[\![.]\!]_{pgs}$ is defined on $PASS$. The corollary follows immediately from the proposition, since $[\![.]\!]_{gs}$ is the restriction of $[\![.]\!]_{pgs}$ to total assignments.

It is also possible to use down-dating to give an elegant definition of dynamic validity, $\models$. First we introduce some notation:

> **Notation**: We will write $\downarrow\langle x\rangle g$ for the assignment f such that $g[\![x\mathsf{E}]\!]f$. For $\downarrow\langle x_n\rangle(\ldots(\downarrow\langle x_1\rangle(g))\ldots)$ we write $\downarrow\langle x_1,\ldots,x_n\rangle(g)$ and if G is a set of partial assignments, we write $\downarrow\langle x_1,\ldots,x_n\rangle(G)$ for $\{\downarrow\langle x_1,\ldots,x_n\rangle(g) : g \in G\}$.

We define:

Definition 5.15 *1. $\ll$ is a relation on states defined by: $(A,S,F) \ll (A',S',F') \Leftrightarrow F \subseteq \downarrow S'(F')$.*

2. For any state (A,S,F), $\downarrow(A,S,F)$, the projection (or domain) of (A,S,F), is defined by: $\downarrow(A,S,F) = (A,\langle\rangle,\downarrow S(F))$.

So to find $\downarrow(A,S,F)$, we simply forget the values of the variables in S. This way the functions in $\downarrow S(F)$ are exactly the ones that have an $\langle\!\langle S\rangle\!\rangle$-extension in F.

Now we have that for any ϕ, $\downarrow[\phi]$ is the input state, or domain, for ϕ: ϕ asks for values on the variables in the first component of $\downarrow[\phi]$ and accepts only those assignments that are in the third component of $\downarrow[\phi]$. $\downarrow S(F_\phi)$ is for $[\phi]$, what $dom([\![\phi]\!]_{gs})$ is for $[\![\phi]\!]_{gs}$.

We claim that 5.15.1 is in fact a definition of dynamic validity. I.e. we claim:

Proposition 5.16 *For any $\phi, \psi : [\phi] \ll [\psi] \Leftrightarrow \phi \models \psi$.*

Proof: By comparing the definition of $\ll$ and, for example, $\models_{[}$, using the fact that $f\langle\!\langle S\rangle\!\rangle g \Leftrightarrow \downarrow S(g) = f$.□

This result seems rather strong, but in fact it is already known for the original DPL-semantics that

$$\phi \models \psi \;\Leftrightarrow\; range([\![\phi]\!]_{gs}) \subseteq dom([\![\psi]\!]_{gs}).$$

Here we see that the same relation holds, but now it is possible to define domains in terms of a more primitive notion: the down-date operator. In the extended language it is even possible to give for each formula an *expression* that gives the domain of the formula: we simply add the right amount of $x\mathsf{E}$'s for each variable. If we call this expression for the domain of ϕ, $\downarrow\phi$, we get for example:

Example:

$\downarrow P(x) = P(x)$;

$\downarrow(\exists x.R(x,y)) = \exists x.R(x,y).x\mathsf{E}$

$\downarrow(\exists x.R(x,y).\exists x.P(x)) = \exists x.R(x,y).\exists x.P(x).x\mathsf{E}.x\mathsf{E}$

The fact that we are able to model down-dating in sequence semantics, shows how powerful sequence semantics really is. In the applications that we have given, we have shown that some familiar notions in the semantics can be redefined elegantly in terms of the down-date operator. It remains to be seen how down-dating can be used in the study of phenomena that are genuinely non-monotonic. Probably $x\mathsf{E}$ could be used to model some cases of belief revision, but that would take us beyond the scope of this paper.

6 Conclusion

The main conclusion of this paper is that it is possible to give a formalization of the ideas of Groenendijk and Stokhof[1991a] in which the information content of a formula can be represented formally. This means that interesting questions about information can be discussed in DPL-semantics.

We have obtained the improved representation by using sequence valued assignments. The use of these assignment inspired a suitable notion of information structure. We discovered that different ways of looking at information in DPL lead to different orderings on the information structures. In further work we hope to improve our understanding of the ways in which information can be compared in dynamic semantics. We think that the general idea that we have developed about comparing information will be of use here.

We were also able to define a down-date operator in our semantics. This operator is an instruction to forget the value of a variable. We have looked at the relation of down-dating with the original semantics for DPL and with the DPL-notion of valid inference. But maybe downdating can also be used to model genuinely non-monotonic phenomena, such as belief revision, that fall ouside the scope of this paper.

7 Acknowledgements

I would like to thank Albert Visser for encouragement and inspiration. Thanks are also due to my colleagues at the O.T.S (Research Institute for Language and Speech) in Utrecht, Holland, and my colleagues from the AIO-network TLI who have provided useful criticism on an earlier version of this paper. I am particularly grateful to Emiel Krahmer for pointing out numerous small errors and typos in the earlier version. I would also like to thank the referees, whose detailed comments have been very helpful.

8 Appendix

In this appendix we present a proof of proposition 2.6 and corollary 2.8. In the proofs we will use the following lemma:

Lemma(Irrelevance):

1. $k[\![\phi]\!]h \Rightarrow k'[\![\phi]\!]h'$ for all k', h' such that for all x $k'(x) = \sigma_x * k(x)$ and $h'(x) = \sigma_x * h(x)$ (for some sequence σ_x).

2. $k[\![\phi]\!]h \Rightarrow k''[\![\phi]\!]h''$ for all k'', h'' such that for all x $k(x) = \sigma_x * k''(x)$ and $h(x) = \sigma_x * h''(x)$ (for some sequence σ_x) and $k''(x)$ extends $end(k(x))$.

The lemma says that if (k,h) is in the relation $[\![\phi]\!]$, only those values of k and h are relevant that occur after $end(k(x))$. This corresponds to the fact that we always add the current value of a variable at the end. ($end(k(x))$ itself is relevant if x is free in ϕ.) We will not prove the lemma: the proof is an easy induction.

Now we will prove proposition 2.6:

Proposition 8.1 *For all $\phi \in DPL$ we have $\Phi([\![\phi]\!]) = [\![\phi]\!]_{pgs}$.*

Remember that Φ is defined by:

Definition 8.2 (2.5) *We define $\Phi : \wp(SASS \times SASS) \to \wp(PASS \times PASS)$ as follows:* $\Phi(R) = \{(g,f) : \exists (k,h) \in R : \forall x \in VAR : f(x) = end(h(x)) \wedge g(x) = end(k(x))\}$.

Proof (of 2.6): We have to prove that:
1 $(g,f) \in \Phi([\![\phi]\!]) \Rightarrow g[\![\phi]\!]_{pgs} f$
2 $g[\![\phi]\!]_{pgs} f \Rightarrow (g,f) \in \Phi([\![\phi]\!]_{pgs})$.

We prove this by a simultaneous induction on the complexity of ϕ. We will need 1 as induction hypothesis for the $\to$-clause of 2 and vice versa.

1. $P(x)$ Suppose $(g,f) \in \Phi([\![P(x)]\!])$. Then there are h,k such that $k = h \wedge end(h(x)) \in I(P) \wedge \forall y : g(y) = end(k(y)) \wedge f(y) = end(h(y))$. Hence $f = g$ and $end(h(x)) = f(x) \in I(P)$, i.e. $g[\![P(x)]\!]_{pgs} f$.

   $\exists x$ Suppose $(g,f) \in \Phi([\![\exists x]\!])$. Then there are h,k $k[\langle x \rangle]h \wedge \forall y : f(y) = end(h(y)) \wedge g(y) = end(k(y))$. Obviously $g[\![\exists x]\!]_{pgs} f$.

   $\psi.\chi$ Suppose $(g,f) \in \Phi([\![\psi.\chi]\!])$. Then there are h,k,l such that: $k[V_\psi * V_\chi]h \wedge h \in F_\chi \wedge l[V_\chi]h \wedge l \in F_\psi \wedge \forall y : g(y) = end(k(y)) \wedge f(y) = end(h(y))$. Define m such that $m(y) = end(l(y))$. Then $(m,f) \in \Phi([\chi])$. Also, since $k[V_\psi]l$, $(g,m) \in \Phi([\psi])$. Hence (ind. hyp.): $g[\![\psi.\chi]\!]_{pgs} f$.

   $(\psi \to \chi)$ Suppose $(g,f) \in \Phi([\![(\psi \to \chi)]\!])$. Then there are h,k such that $k[\![(\psi \to \chi)]\!]h \wedge \forall y : g(y) = end(k(y)) \wedge f(y) = end(h(y))$. Then $f = g$, $k = h$. Now let $f[\![\psi]\!]_{pgs} m$. Then 2 gives us $l[\![\psi]\!]n$. Now the lemma gives us n' such that $h[\![\psi]\!]n'$. By assumption this means that there is a n'' $n'[\![\chi]\!]n''$. By the definition of Φ, this gives a p such that $(m,p) \in \Phi([\![\chi]\!])$. Now the induction hypothesis for 1 guarantees $m[\![\chi]\!]_{pgs} p$. Hence $g[\![(\psi \to \chi)]\!]_{pgs} f$.

2. $P(x)$ Suppose $g[\![P(x)]\!]_{pgs} f$. Then $g = f \wedge end(f(x)) \in I(P)$. So we can choose $h = k = f$ to prove that $(g,f) \in \Phi([\![P(x)]\!])$.

   $\exists x$ Suppose $g[\![\exists x]\!]_{pgs} f$. Then $\forall y : (g(y) = f(y) \wedge (y = x \wedge f(x)$ is defined)). Choose h such that $h(x) = g(x) * \langle f(x) \rangle$ and $h(y) = f(y)$ for all other y, and choose k such that $k(x) = pd(h(x))$ and $k(y) = h(y)$ for all other y. These h,k guarantee that $(g,f) \in \Phi([\![\exists x]\!])$.

   $(\psi \to \chi)$ Suppose $g[\![(\psi \to \chi)]\!]_{pgs} f$. Then $f = g \wedge \forall m : g[\![\psi]\!]_{pgs} m \ \exists n : m[\![\chi]\!]_{pgs} n$. Let $k = f = g = h$. We prove that $k[\![(\psi \to \chi)]\!]h$: suppose $k[\![\psi]\!]l$. This gives $(g,p) \in \Phi([\![\psi]\!])$ for p such that $p(y) = end(l(y))$ for all y. Hence, by 1, $g[\![\psi]\!]_{pgs} p$. But then, by assumption, there must be a q such that $p[\![\chi]\!]_{pgs} q$. By induction hypothesis for 2 we get h', l' such that $l'[\![\chi]\!]h' \wedge \forall y : end(h'(y)) = q(y) \wedge end(l'(y)) = p(y) = end(l(y))$. Using the lemma we find that there is a h'' such that $l[\![\chi]\!]h''$.

$\psi.\chi$ Suppose $g[\![\psi.\chi]\!]_{pgs}f$. It suffices to consider the cases in which χ is not of the form $\chi'.\chi''$.

a $\chi \equiv P(x)$: Then $g[\![\psi]\!]_{pgs}f \ \wedge \ f(x) \in I(P)$. By induction hypothesis for 2 we get h,k such that $k[\![\psi]\!]h \ \wedge \ \forall y : f(y) = end(h(y)) \ \wedge \ g(y) = end(k(y))$. But then $end(h(x)) \in I(P)$. So these h,k also do the job for $\psi.P(x)$.

b $\chi \equiv \exists x$: Then there is a f' such that f' does not differ from f on variables other than x and $g[\![\psi]\!]_{pgs}f'$. By the induction hypothesis for 2, there are k,h' such that $k[\![\psi]\!]h' \ \wedge \ \forall y : end(h'(y)) = f'(y) \ \wedge \ end(k(y)) = g(y)$. Now we define h such that $h(x) = h'(x) * \langle f(x) \rangle$ and $h(y) = h'(y)$ for all other variables. Then h,k do the job for $\psi.\exists x$.

c $\chi \equiv (\rho \rightarrow \tau)$: Then $g[\![\psi]\!]_{pgs}f \ \wedge \ f[\![(\rho \rightarrow \tau)]\!]_{pgs}f$. By the induction hypothesis for 2 there are k,h,h' such that $k[\![\psi]\!]h \ \wedge \ h'[\![(\rho \rightarrow \tau)]\!]h'$, where for all x $end(h(x)) = end(h'(x)) = f(x)$. By the lemma we find that also $h[\![(\rho \rightarrow \tau)]\!]h$. But then $(g,f) \in \Phi([\![\phi]\!])$. □

Note that the proof of the proposition is not difficult. It is just hard work. The same holds for the proof of corollary 2.8. This time we will provide less details.

Proof (corollary 2.8): We sketch the proof of $1 \Leftrightarrow 2$ and leave $2 \Leftrightarrow 3$ to the reader.

$1 \Rightarrow 2$ Assume $\phi \models_{[\![}\ \psi$. Let $f,g \in PASS$ be given such that $f[\![\phi]\!]_{pgs}g$. We have to find $h \in PASS$ such that $g[\![\psi]\!]_{pgs}h$. Now, since $PASS \subseteq SASS$, it follows from the assumption that we find a $h' \in SASS$ such that $g[\![\psi]\!]h'$. But then Φ gives a $h \in PASS$ such that $g\Phi([\![\psi]\!])h$, i.e. $g[\![\psi]\!]_{pgs}h$.

$2 \Rightarrow 1$ Assume $\phi \models_{pgs} \psi$. Let $f,g \in SASS$ be given such that $f[\![\phi]\!]g$. We have to find $h \in SASS$ such that $g[\![\psi]\!]h$. Φ gives for f,g $f',g' \in PASS$ such that $f'[\![\phi]\!]_{pgs}g'$. By the assumption this gives an $h' \in PASS$ such that $g'[\![\psi]\!]_{pgs}h'$. If we decompose ψ into atoms $\psi_0,\ldots,\psi_n$, then we can see all the changes of values that we need to build the $h \in SASS$ such that $g[\![\psi]\!]h$.□

9 References

Benthem, J. van "Semantic parallels in natural language and computation", in: H. Ebbinghaus et al. (eds.), *Logic Colloquium '87*, Amsterdam,1989

Benthem, J. van "General Dynamics", *Theoretical Linguistics*, pp. 159-201, 1991

Dekker, P. "An update semantics for dynamic predicate logic", to appear in: P. Dekker, M. Stokhof (eds.) *Proceedings of the* 8^{th} *Amsterdam Colloquium*, 1992

Fernando, T. *Transition Systems*, manuscript, 1991 (to be presented at JELIA 1992, Berlin)

Groenendijk, J. and M. Stokhof "Dynamic Predicate Logic", *Linguistics and Philosophy 14*, pp. 39-100, 1991a

Groenendijk, J. and M. Stokhof "Two theories of dynamic semantics", in: J. van Eijck (ed.), *Logics in AI — European Workshop JELIA '90*, pp. 55-64, Berlin, 1991b

Kamp, H. "A Theory of Truth and Semantic Representation", in: Groenendijk et al. eds., *Truth, Interpretation and Information*, Dordrecht, 1984

Kamp, H. and U. Reyle *From Discourse to Logic*, manuscript, Institute for Computational Linguistics, University of Stuttgart, 1990

Pratt, V. "Action Logic and Pure Induction" in: J. van Eijck (ed.), *Logics in AI — European Workshop JELIA '90*, pp. 55-64, Berlin, 1991

Veltman, F. "Defaults in update semantics", manuscript, Department of Philosophy, University of Amsterdam, 1991 (to appear in: *Journal of Philosophical Logic*)

Visser, A. *Actions under Presupposition* in: Logic Group Preprint Series, no.76, 1992

Zeevat, H. "A Compositional Approach to Discourse Representation Theory", *Linguistics and Philosophy 12*, pp.95-131