Information and Computation 207 (2009) 980–999

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Situations in LTL as strings[☆]

Tim Fernando

Computer Science Department, Trinity College, Dublin 2, Ireland

ARTICLE INFO

Article history:
Received 21 November 2006
Revised 15 May 2008
Available online 11 April 2009

Keywords:
Situation
LTL
String
Natural language semantics

ABSTRACT

Situations serving as worlds as well as events in linguistic semantics are formulated as strings recording observations over discrete time. This formulation is applied to a linear temporal logic, in line with L. Schubert's distinction between described and characterized situations. The distinction is developed topologically and computationally, and linked to the opposition between truth-conditional and proof-conditional semantics. For a finitary handle on quantification, strings are associated with situations not only on the basis of observation but also through derivation and constraint satisfaction. The constraints specified lead to an implementation simpler than the derivations given.

1. Introduction

Situations have over the years attracted varying degrees of attention in natural language semantics ([23,2,16] and many others). They have been called on to do the work of worlds for modality, and also the work of events in temporality. An important distinction between these uses can be put in terms introduced by Schubert [24] as that between description and characterization. To say that a formula φ *describes* a situation/world s is to say that φ is true in s. To say that φ *characterizes* a situation/event s is to demand in addition that s be (in some sense) *about* φ (so that φ is true *of* s and not just *in* s). For φ to characterize s, φ and s must fit tightly, with *no* part of s extraneous to φ. Schubert argues that this tight fit is essential if φ *because* φ' is to be analyzed in terms of a causal relation between situations s and s' that φ and φ', respectively, characterize.[1] In general, a world is too big to be characterized by a formula, especially a tautology, relative to which every bit of a world is arguably extraneous. A less extreme example from natural language is a formula $\hat{\varphi}$ saying Pat walks from the office to the train station. A situation characterized by $\hat{\varphi}$ ought not to include any subsequent train rides. This suggests using a "part of" relation $\sqsubseteq$ on situations to express the additional requirement characterization imposes on description. For instance, we might reduce the claim that a formula φ characterizes a situation s to the description and minimality condition in (†).

(†) φ describes s, and φ describes no other situation s' such that $s' \sqsubseteq s$

Such a reduction would imply (‡).

(‡) whenever φ characterizes s, φ characterizes no other situation s'
such that $s' \sqsubseteq s$

[☆] A preliminary version of this paper appeared in *Electronic Notes in Theoretical Computer Science*, vol. 165, 22 November 2006, pp. 23–36 (Proceedings of the 13th Workshop on Logic, Language, Information and Computation, 2006). I am grateful for this opportunity to expand the article, and thank the editors and referees for their help.

E-mail address: Tim.Fernand@tcd.ie
URL: http://www.cs.tcd.ie/Tim.Fernando/

[1] Causation is just one of many relations between situations characterized by discourse segments related by rhetorical (or coherence) relations. These situations are called *main eventualities* in Asher and Lascarides [1], and underpin rhetorical relations such as *Elaboration* and *Narration*, as well as *Explanation*.

0890-5401/$ - see front matter © 2009 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2008.11.003

Although (‡) poses no obvious problem for say, an event of Pat walking from the office to the train station, difficulties arise when the modification "from the office to the train station" is dropped: it is widely accepted that an event of Pat walking may have proper parts that count as Pat walking. (Further complications with minimization are discussed, for instance, in Dekker [6].)

1.1. Characterization as the basis of description

Rather than reducing characterization to description, we may derive description from characterization of a $\sqsubseteq$-part, as in (1), for a wide class of formulas φ.

$$\varphi \text{ describes } s \iff (\exists s' \sqsubseteq s)\ \varphi \text{ characterizes } s' \tag{1}$$

Formulas φ for which (1) hold are *persistent* insofar as an immediate consequence of (1) and the transitivity of $\sqsubseteq$ is the implication (2).

$$\varphi \text{ describes } s' \text{ and } s' \sqsubseteq s \text{ implies } \varphi \text{ describes } s \tag{2}$$

Counter-examples include the formula φ saying *p throughout every described time*, for which it is appropriate to assert

$$\varphi \text{ describes } s \iff \varphi \text{ describes } \textit{every} \sqsubseteq\text{-part of } s$$

rather than

$$\varphi \text{ describes } s \iff \varphi \text{ describes } \textit{some} \sqsubseteq\text{-part of } s.$$

That said, we opt for the latter equivalence in what follows, studying formulas for which truth was originally defined relative to worlds. Worlds are $\sqsubseteq$-maximal situations, making (2) vacuously true for worlds s'. Thus, if we restrict description to worlds, and understand s in the equivalence (1) to range over worlds, we need not worry that *p throughout every described time* violates (2). (It can't.) Better yet, we might make do with its variant *p throughout every characterized time*, which is unproblematic, as (2) fails for "characterize" in place of "describe".

1.2. Application to time: strings

The notion of characterization is fleshed out below against formulas from a linear temporal logic under the assumption that next and previous moments are well-defined. In defense of this assumption, Prior writes

> The usefulness of systems of this sort does not depend on any serious metaphysical assumption that time *is* discrete; they are applicable in limited fields of discourse in which we are concerned with what happens in a sequence of discrete states, e.g. in the workings of a digital computer ([21], p. 67).

The distance to the previous/next moment is a matter of granularity, which we leave open and refinable by any finite number of observations. Accordingly, the Priorean formula Past p saying that p held sometime in the past may be pictured as the regular language

$$\boxed{p}\,\square^*\,\boxed{now} = \boxed{p}\boxed{now} + \boxed{p}\,\square\,\boxed{now} + \boxed{p}\,\square\square\,\boxed{now} + \cdots \tag{3}$$

consisting of strings $\boxed{p}\,\square^n\,\boxed{now}$ of length $n+2$ (for $n \geq 0$), with substring $\square^n$ representing n intervening moments between the observation of p and the present (*now*). Reversing these strings, we get the language

$$\boxed{now}\,\square^*\,\boxed{p} = \boxed{now}\boxed{p} + \boxed{now}\,\square\,\boxed{p} + \boxed{now}\,\square\square\,\boxed{p} + \cdots \tag{4}$$

for the formula Future p saying that in the future p holds. Each of the strings in (3) and (4) is a sequence of snapshots that can be viewed as an event (e.g. Tenny [26]). In contrast to the situations in McCarthy and Hayes [18], the snapshots here need not be an exhaustive record of (atomic) propositions true at that moment. This partiality allows the languages in (3) and (4) to be understood as the events characterized by Past p and Future p, respectively. This is made precise in Section 2, where strings are reformulated as schedules and structured topologically by $\sqsubseteq$.

1.3. From observations to derivations and constraints

An obvious challenge to the finite-state (regular language) approach sketched above is the Priorean formula Future$_\forall$ p saying that p is true every moment after the present, pictured in (5).

$$\boxed{now}\boxed{p}^{\infty} = \lim_{n\to\infty} \boxed{now}\boxed{p}^{n} \tag{5}$$

The computational complications posed by (5) and other temporal formulas (both more and less complex than $\text{F}\textsc{uture}_{\forall}\, p$) motivate a step from strings-as-observations to strings-as-constructions. Very briefly, the disjunctive normal form in a subbasis/basis presentation of topologies makes way for the higher types of proof-theoretic semantics. The progressively longer (finite) strings approximating the infinite string in (5) are derived in Section 3 from incremental replace rules that can be implemented by finite-state transducers. Finite-state constraints that capture sufficient iterated applications of these rules are studied in Section 4, constituting a declarative alternative to the procedural methods of Section 3. We step back in Section 5 and conclude.

1.4. *Events in linguistic semantics*

Before proceeding to the technical developments in Sections 2–5, let us pause for some more motivation. Temporal logic and event semantics have competed since the days of Prior and Reichenbach for the attention of linguists. These approaches can be brought together under the view of situations-as-strings, applied to linear temporal logic below, having been tried out previously against

(a) the analysis in Reichenbach [23] of tense and aspect, with *event time* strung out, and formulas S and R for speech time and reference time such that, for instance, *It rained from dawn to dusk* is represented by the regular language

$$\boxed{\text{rain,dawn}}\boxed{\text{rain}}^{*}\boxed{\text{rain,dusk},R}\square^{*}\boxed{S}$$

and the difference between

Pat left Dublin but is back

and

Pat has left Dublin ?but is back

is accounted for in terms of inertia ([8])

(b) the classification (going back to Vendler [27]) of events based on telicity and the difference between temporal *for* and *in* modification ([7])

(c) Kamp's event structures ([15]), which are represented in [12] as projective limits of strings, and

(d) so-called anankastic conditionals ([9]).

Strings-as-situations are compared to the *Event Calculus* of van Lambalgen and Hamm [17] in [11], and extended with branching to cover modal implications of the progressive in [13].

2. Schedules and linear temporal logic

It will prove convenient to

(i) identify time with the set $\mathbb{Z}$ of positive and non-positive integers,[2] reserving 0 to mark out the present, and to

(ii) push negation towards the atoms, fixing a set P of propositions with a map $\bar{\cdot} : P \to P$ such that for each $p \in P, \overline{p} \neq p$ but $\overline{\overline{p}} = p$.

Let us refer to a relation $s \subseteq \mathbb{Z} \times P$ between integers and propositions in P as a *P-schedule*, the idea being that $s(i,p)$ says p is true at time i. For example, the string $\boxed{p}\square\square\boxed{now}\boxed{p,q}$ corresponds to the P-schedule $\{(-3,p),(1,p),(1,q)\}$, for $P \supseteq \{p,q\}$ with *now* understood to be at 0. Let us call a finite P-schedule a *P-strip* (as it can be written as a string in $\mathsf{Pow}(P \cup \{\text{now}\})^{*}$) and write Str_P for the set

$$Str_P = \{s \subseteq \mathbb{Z} \times P \mid s \text{ has finite cardinality}\}$$

of P-strips. A *P-world* is a P-schedule w such that for all $p \in P$ and $i \in \mathbb{Z}$,

$$w(i,p) \iff \text{not } w(i,\overline{p}).$$

A *P-situation* is a P-schedule s such that $s \subseteq w$, for some P-world w. (In other words, a P-situation is a P-schedule s such that for *no* i,p is it the case that $s(i,p)$ and $s(i,\overline{p})$.) We equate the part-of relation $\sqsubseteq$ with the subset relation $\subseteq$, and drop

[2] We relax this identification in Section 5 to consider, for instance, the real line.

the modification P- on schedules, situations, strips and worlds when we can. This section concerns topologies over sets $X \subseteq \mathsf{Pow}(\mathbb{Z} \times P)$ of schedules induced by sets of described schedules from a notion of characterization according to (1). Typically (but not necessarily), the schedules in X will be worlds,[3] and a characterized situation will be a strip s, understood as a finite set of observations of any $x \in X$ such that $s \subseteq x$.

2.1. Topologies based on finite observations

Applying the notion of characterization to a set A of schedules in place of a formula, let us agree to read "$s \in A$" as *A characterizes s*, so that under (1), we have

$$A \text{ describes } x \iff (\exists s \subseteq x)\, s \in A.$$

Next, given a set $X \subseteq \mathsf{Pow}(\mathbb{Z} \times P)$ of schedules, we define

(i) $X_A \subseteq X$ to be the set of schedules in X described by A

$$X_A = \{x \in X \mid (\exists s \subseteq x)\, s \in A\}$$

(ii) $Str(X) \subseteq Str_P$ to be the set of strips $\subseteq$-contained in schedules in X

$$Str(X) = \{s \in Str_P \mid (\exists x \in X)\, s \subseteq x\}$$

and

(iii) $\mathcal{O}(X) \subseteq \mathsf{Pow}(X)$ to be the family of subsets of X determined by sets $A \subseteq Str(X)$ of finite observations

$$\mathcal{O}(X) = \{X_A \mid A \subseteq Str(X)\}.$$

Proposition 1. *Let $X \subseteq \mathsf{Pow}(\mathbb{Z} \times P)$.*

(a) $\mathcal{O}(X) = \{X_A \mid A \subseteq Str_P\}$ *and indeed for all $A \subseteq Str_P$, $X_A = X_{A \cap Str(X)}$.*
(b) $(X, \mathcal{O}(X))$ *is a subspace of the topological space* $(\mathsf{Pow}(\mathbb{Z} \times P), \mathcal{O}(\mathsf{Pow}(\mathbb{Z} \times P)))$.
(c) *For all $A, B \subseteq \mathsf{Pow}(\mathbb{Z} \times P)$,*

$$X_A \cup X_B = X_{A \cup B}$$
$$X_A \cap X_B = X_{A \sqcap B}$$

where $A \sqcap B$ is defined to be $\{s \cup r \mid s \in A \text{ and } r \in B\}$.

The topological space $(X, \mathcal{O}(X))$ foregrounds the "reality" $x \in X_A \in \mathcal{O}(X)$ that is presumably observed, at the expense of the finite observations $s \in A \subseteq Str(X)$ that shape $\mathcal{O}(X)$. An alternative that downplays the "points" $x \in X$ and highlights the observations is to endow $\mathsf{Pow}(Str(X))$ with the structure of a *locale*, consisting, under the presentation in Vickers [28], of

(i) binary operations $\sqcup_X$ and $\sqcap_X$ given by

$$A \sqcup_X B = A \cup B$$
$$A \sqcap_X B = (A \sqcap B) \cap Str(X)$$

for all $A, B \subseteq Str(X)$, and

(ii) the set $\{\hat{y} \mid y \subseteq \mathbb{Z} \times P\}$ of functions $\hat{y} : \mathsf{Pow}(Str(X)) \to \{0, 1\}$ given by a schedule y such that for all $A \subseteq Str(X)$,

$$\hat{y}(A) = 1 \iff (\exists s \in A)\, s \subseteq y.$$

A notion halfway between topological space and locale is that of a *topological system* $(X, \mathsf{Pow}(Str(X)))$ with satisfaction relation $\models\, \subseteq X \times \mathsf{Pow}(Str(X))$ defined by

$$x \models A \iff x \in X_A$$

for all $x \in X$ and $A \subseteq Str(X)$. From the topological system $(X, \mathsf{Pow}(Str(X)))$, Vickers derives the topological space $(X, \mathcal{O}(X))$ by *spatialization* and the locale $(\{\hat{y} \mid y \subseteq \mathbb{Z} \times P\}, \mathsf{Pow}(Str(X)))$ by *localification*. For our present purposes, the situation

[3] The reader is free to assume that the schedules in X are worlds, and write W and w in place of X and x below. But that assumption is not necessary for any of the arguments made.

Table 1
Vickers [28] applied to the situation in Schubert [24].

Vickers		Schubert
Topological system $(X, \mathrm{Pow}(Str(X)))$	Situation	
Topological space $(X, \{X_A \mid A \subseteq Str(X)\})$	Point/world x	Describe (X_A)
locale $(\{\hat{y} \mid y \subseteq \mathbb{Z} \times P\}, \mathrm{Pow}(Str(X)))$	Strip/event s	Characterize (A)

is summarized in Table 1. The remainder of this paper is directed at developing the "pointless topology"-perspective of locale theory for the concrete case of a linear temporal logic (extending Priorean tense logic with Kamp's *since* and *until* constructs) putting the set Str_P of strips in center stage, and consigning the particular choice X of points to the background. That said, we proceed from the point/world-oriented semantics traditionally given to linear temporal logic (under Kripke semantics).

2.2. Temporal formulas and characterization

Given a set $P_\circ$ of atomic propositions, let

(i) $P = P_\circ \times \{+, -\}$, and for $a \in P_\circ$, $\overline{(a,+)} = (a,-)$ and $\overline{(a,-)} = (a,+)$, and
(ii) Φ be the set of temporal formulas φ generated from $p \in P$ by

$$\varphi ::= p \mid \top \mid \bot \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \textsc{Next}\ \varphi \mid \textsc{Previous}\ \varphi \mid \\ \varphi\ \textsc{until}\ \varphi \mid \varphi\ \textsc{since}\ \varphi \mid \varphi\ \textsc{release}\ \varphi \mid \varphi\ \textsc{initiate}\ \varphi.$$

Building a Kripke frame around $\mathbb{Z}$, we can interpret each formula $\varphi \in \Phi$ relative to a function (valuation) $v : \mathbb{Z} \to \mathrm{Pow}(P_\circ)$ and integer $i \in \mathbb{Z}$ in the usual way, with

$$v, i \models p \iff x_v(i, p)$$

where x_v is the P-world

$$\{(i, (a, +)) \mid i \in \mathbb{Z},\ a \in v(i)\} \cup \{(i, (a, -)) \mid i \in \mathbb{Z},\ a \in P_\circ - v(i)\}$$

and $v, i \models \top$, $v, i \not\models \bot$,

$$\begin{aligned}
v, i \models \varphi \wedge \psi &\iff v, i \models \varphi \text{ and } v, i \models \psi \\
v, i \models \varphi \vee \psi &\iff v, i \models \varphi \text{ or } v, i \models \psi \\
v, i \models \textsc{Next}\ \varphi &\iff v, i + 1 \models \varphi \\
v, i \models \textsc{Previous}\ \varphi &\iff v, i - 1 \models \varphi \\
v, i \models \varphi\ \textsc{until}\ \psi &\iff (\exists j \geq i)\ (v, j \models \psi \text{ and} \\
&\qquad\qquad \text{for } i \leq k < j,\ v, k \models \varphi) \\
v, i \models \varphi\ \textsc{since}\ \psi &\iff (\exists j \leq i)\ (v, j \models \psi \text{ and} \\
&\qquad\qquad \text{for } j < k \leq i,\ v, k \models \varphi) \\
v, i \models \varphi\ \textsc{release}\ \psi &\iff (\forall j \geq i)\ (v, j \models \psi \text{ or} \\
&\qquad\qquad (\exists k \geq i)(k < j \text{ and } v, k \models \varphi)) \\
v, i \models \varphi\ \textsc{initiate}\ \psi &\iff (\forall j \leq i)\ (v, j \models \psi \text{ or} \\
&\qquad\qquad (\exists k \leq i)(k > j \text{ and } v, k \models \varphi)).
\end{aligned}$$

Note that we can construe negation as an extension of $\bar{\cdot} : P \to P$ to Φ along the dual pairs $(\top, \bot)$, $(\wedge, \vee)$, $(\textsc{Next}, \textsc{Next})$, $(\textsc{Previous}, \textsc{Previous})$, $(\textsc{until}, \textsc{release})$ and $(\textsc{since}, \textsc{initiate})$ so that e.g.

$$\overline{\varphi \wedge \psi} = \overline{\varphi} \vee \overline{\psi}$$
$$\overline{\varphi \vee \psi} = \overline{\varphi} \wedge \overline{\psi}.$$

Also, we can equate $\textsc{Future}\ \varphi$ with $\top$ $\textsc{until}$ $\textsc{Next}\ \varphi$, and $\textsc{Future}_\forall\ \varphi$ with $\bot$ $\textsc{release}$ $\textsc{Next}\ \varphi$, etc.

As $\models$ specifies what P-worlds formulas in Φ describe, the next question is: what about characterized events? For such a notion, a minimal condition of faithfulness to $\models$ is the following. Given a set $A \subseteq \mathrm{Pow}(\mathbb{Z} \times P)$ of schedules, an integer $i \in \mathbb{Z}$, and a formula $\varphi \in \Phi$, we say A *i-portrays* φ if for every function $v : \mathbb{Z} \to \mathrm{Pow}(P_\circ)$,

$$v, i \models \varphi \iff (\exists s \subseteq x_v)\, s \in A.$$

The aim of this subsection is to define for every pair $(i, \varphi) \in \mathbb{Z} \times \Phi$, a set $[\![i, \varphi]\!]$ of schedules satisfying Proposition 2.

Proposition 2. *Let $\varphi \in \Phi$ and $i \in \mathbb{Z}$.*

(a) $[\![i, \varphi]\!]$ *i-portrays φ.*
(b) *If neither* RELEASE *nor* INITIATE *occurs in φ, then $[\![i, \varphi]\!] \subseteq Str_P$.*

We define $[\![i, \varphi]\!]$ by induction on φ, with the base case $[\![i, \bot]\!] = \emptyset$, $[\![i, \top]\!] = \{\emptyset\}$ and $[\![i, p]\!] = \{\{(i, p)\}\}$. Inductively,

$$\begin{aligned}
[\![i, \varphi \wedge \psi]\!] &= [\![i, \varphi]\!] \sqcap [\![i, \psi]\!] \quad (= \{s \cup r \mid s \in [\![i, \varphi]\!],\ r \in [\![i, \psi]\!]\}) \\
[\![i, \varphi \vee \psi]\!] &= [\![i, \varphi]\!] \cup [\![i, \psi]\!] \\
[\![i, \text{NEXT}\ \varphi]\!] &= [\![i + 1, \varphi]\!] \\
[\![i, \text{PREVIOUS}\ \varphi]\!] &= [\![i - 1, \varphi]\!].
\end{aligned}$$

As for UNTIL, guided by the picture of $[\![0, p \text{ UNTIL } q]\!]$ as

$$\boxed{now, q} + \boxed{now, p}\boxed{p}^{*}\boxed{q},$$

we set

$$[\![i, \varphi \text{ UNTIL } \psi]\!] = \bigcup_{n \geq 0} (u(\varphi, n, i) \sqcap [\![i + n, \psi]\!])$$

where $u(\varphi, n, i)$ is defined by induction on $n \geq 0$ to witness φ between i and $i + n - 1$: $u(\varphi, 0, i) = \{\emptyset\}$ and

$$u(\varphi, n + 1, i) = [\![i, \varphi]\!] \sqcap u(\varphi, n, i + 1).$$

Temporally reversing UNTIL to get SINCE, let

$$[\![i, \varphi \text{ SINCE } \psi]\!] = \bigcup_{n \geq 0} (s(\varphi, n, i) \sqcap [\![i - n, \psi]\!])$$

where $s(\varphi, 0, i) = \{\emptyset\}$ and

$$s(\varphi, n + 1, i) = [\![i, \varphi]\!] \sqcap s(\varphi, n, i - 1).$$

Turning to RELEASE, let us consider first the special case $\bot$ RELEASE φ, noting that $[\![i, \bot \text{ RELEASE } \varphi]\!]$ is roughly the limit of $u(\varphi, n, i)$ as $n \to \infty$. More precisely, we put

$$[\![i, \bot \text{ RELEASE } \varphi]\!] = \left\{ \bigcup_{n \geq 0} f(n) \mid f \in F_{+}(\varphi, i) \right\}$$

where $F_{+}(\varphi, i)$ is the set of functions $f : \mathbb{N} \to \text{Pow}(\mathbb{Z} \times P)$ such that for all $n \geq 0, f(n) \in [\![i + n, \varphi]\!]$. Then set

$$[\![i, \varphi \text{ RELEASE } \psi]\!] = [\![i, \psi \text{ UNTIL } (\varphi \wedge \psi)]\!] \ \cup\ [\![i, \bot \text{ RELEASE } \psi]\!].$$

Similarly, for INITIATE, let $F_{-}(\varphi, i)$ be the set of functions $f : \mathbb{N} \to \text{Pow}(\mathbb{Z} \times P)$ such that $f(n) \in [\![i - n, \varphi]\!]$ for all $n \geq 0$, and set

$$\begin{aligned}
[\![i, \bot \text{ INITIATE } \varphi]\!] &= \left\{ \bigcup_{n \geq 0} f(n) \mid f \in F_{-}(\varphi, i) \right\} \\
[\![i, \varphi \text{ INITIATE } \psi]\!] &= [\![i, \psi \text{ SINCE } (\varphi \wedge \psi)]\!] \ \cup\ [\![i, \bot \text{ INITIATE } \psi]\!].
\end{aligned}$$

2.3. Schedules from sequences of strips

For $p \in P$, the single schedule $\{(n, p) \mid n \geq 0\}$ in $[\![0, \bot \text{ RELEASE } p]\!]$ can be pictured topologically

$$\boxed{now, p}\boxed{p}^{\infty} = \lim_{n \to \infty} \boxed{now, p}\boxed{p}^{n}$$

over the space $(X, \mathcal{O}(X))$,[4] provided X has enough points—e.g. if X contains all P-worlds. By contrast, it is noteworthy that $[\![i,\varphi]\!]$ is defined independently of the choice X of points.

In general, we can reformulate the sets $[\![i,\varphi]\!] \subseteq \mathsf{Pow}(\mathbb{Z} \times P)$ of schedules as sets $\langle\!\langle i,\varphi\rangle\!\rangle \subseteq \mathbb{N} \to \mathit{Str}_P$ of functions from $\mathbb{N}$ to Str_P such that

$$[\![i,\varphi]\!] = \left\{ \bigcup_{n\geq 0} f(n) \mid f \in \langle\!\langle i,\varphi\rangle\!\rangle \right\} \tag{6}$$

as follows. We

(i) turn $A \subseteq \mathit{Str}_P$ into $\{\lambda n.s \mid s \in A\} \subseteq \mathbb{N} \to \mathit{Str}_P$ where $\lambda n.s$ is the constant function mapping every $n \in \mathbb{N}$ to s
(ii) redefine $\sqcap$ on $\mathsf{A}, \mathsf{B} \subseteq \mathbb{N} \to \mathit{Str}_P$ by

$$\mathsf{A} \sqcap \mathsf{B} = \{\lambda n.(f(n) \cup g(n)) \mid f \in \mathsf{A} \text{ and } g \in \mathsf{B}\}$$

and
(iii) apply projection functions π, π' of a pairing function on $\mathbb{N}$ to define

$$\langle\!\langle i, \bot \text{ RELEASE } \varphi\rangle\!\rangle = \{\lambda n.h(\pi(n), \pi'(n)) \mid h \in \mathsf{H}_+(i,\varphi)\}$$

where $\mathsf{H}_+(i,\varphi)$ consists of functions $h : \mathbb{N} \times \mathbb{N} \to \mathit{Str}_P$ such that for $n \geq 0$,

$$\lambda m.h(n,m) \in \langle\!\langle i+n,\varphi\rangle\!\rangle$$

(and similarly for $\langle\!\langle i, \bot \text{ INITIATE } \varphi\rangle\!\rangle$, with $\lambda m.h(n,m) \in \langle\!\langle i-n,\varphi\rangle\!\rangle$).

Steps (i) and (ii) lift Str_P to $\mathbb{N} \to \mathit{Str}_P$, while step (iii) keeps us from ascending further to $\mathbb{N} \to (\mathbb{N} \to \mathit{Str}_P)$ and beyond, by Currying and pairing

$$\mathbb{N} \to (\mathbb{N} \to \mathit{Str}_P) \cong (\mathbb{N} \times \mathbb{N}) \to \mathit{Str}_P \cong \mathbb{N} \to \mathit{Str}_P.$$

To bring out the subbasis $\{[\![i,p]\!] \mid p \in P\}$ for $(X, \mathcal{O}(X))$, we can reduce the functions $f : \mathbb{N} \to \mathit{Str}_P$ in $\langle\!\langle i,\varphi\rangle\!\rangle$ further to partial functions $f : \mathbb{N} \rightharpoonup (\mathbb{Z} \times P)$ to $\mathbb{Z} \times P$ by re-analyzing $\mathsf{A} \sqcap \mathsf{B}$ as $\{f \wedge g \mid f \in \mathsf{A} \text{ and } g \in \mathsf{B}\}$ with

$$\begin{aligned}(f \wedge g)(2n) &\simeq f(n)\\ (f \wedge g)(2n+1) &\simeq g(n)\end{aligned}$$

for $n \geq 0$. There is little to choose between encoding a P-schedule as

$$\{f(n) \mid n \in \mathsf{domain}(f)\} \text{ for some partial function } f : \mathbb{N} \rightharpoonup (\mathbb{Z} \times P)$$

or as $\bigcup_{n\geq 0} f(n)$ for $f : \mathbb{N} \to \mathit{Str}_P$. For a canonical system of approximations relative to an enumeration $P = \{p_n\}_{n\geq 0}$ of P (which may repeat in case P is finite), we let S_n, for $n \geq 0$, be the set of $\{p_m \mid m \leq n\}$-strips with domain restricted to integers whose absolute value is $\leq n$

$$S_n = \mathsf{Pow}(\{i \in \mathbb{Z} \mid |i| \leq n\} \times \{p_m \mid m \leq n\})$$

and define functions $\mathsf{r}_n : S_{n+1} \to S_n$ mapping $s \in S_{n+1}$ to its intersection with $\{i \in \mathbb{Z} \mid |i| \leq n\} \times \{p_m \mid m \leq n\}$

$$\mathsf{r}_n(s) = \{(i, p_m) \in s \mid |i| \leq n \text{ and } m \leq n\}.$$

The inverse limit of the projections $(\mathsf{r}_n)_{n\geq 0}$

$$\varprojlim S_n = \left\{ (s_n)_{n\geq 0} \in \prod_{n\geq 0} S_n \;\middle|\; s_n = \mathsf{r}_n(s_{n+1}) \text{ for all } n \geq 0 \right\}$$

is isomorphic to the full set $\mathsf{Pow}(\mathbb{Z} \times P)$ of P-schedules, with $(s_n)_{n\geq 0}$ corresponding to $\bigcup_{n\geq 0} s_n$, and conversely, $x \subseteq \mathbb{Z} \times P$ decomposing to $(\{(i,p_m) \in x \mid |i| \leq n \text{ and } m \leq n\})_{n\geq 0}$. But whereas the sets $\langle\!\langle i,\varphi\rangle\!\rangle$ above can be used to define $[\![i,\varphi]\!]$ according to (6), it is not clear how to construct the inverse limit projections of $[\![i,\varphi]\!]$ without first constructing $[\![i,\varphi]\!]$, given that future and past operators may scope over each other in φ. (More below, especially Section 5.1.)

[4] Recall that x is a *limit point* of a set $A \subseteq X$ if every open set containing x intersects $A - \{x\}$.

Table 2
Replace rules for $\varphi, \psi \in \Phi$.

$\boxed{\varphi \vee \psi}$	$\rightsquigarrow$	$\boxed{\varphi}$	$\boxed{\varphi \vee \psi}$	$\rightsquigarrow$	$\boxed{\psi}$
$\boxed{\textsc{Next}\ \varphi}\Box$	$\rightsquigarrow$	$\Box\boxed{\varphi}$	$\Box\boxed{\textsc{Previous}\ \varphi}$	$\rightsquigarrow$	$\boxed{\varphi}\Box$
$\boxed{\varphi \wedge \psi}$	$\rightsquigarrow$	$\boxed{\varphi, \psi}$	$\boxed{\top}$	$\rightsquigarrow$	$\Box$
$\boxed{\varphi\ \textsc{until}\ \psi}$	$\rightsquigarrow$	$\boxed{\psi}$	$\boxed{\varphi\ \textsc{until}\ \psi}\Box$	$\rightsquigarrow$	$\boxed{\varphi}\boxed{\varphi\ \textsc{until}\ \psi}$
$\boxed{\varphi\ \textsc{since}\ \psi}$	$\rightsquigarrow$	$\boxed{\psi}$	$\Box\boxed{\varphi\ \textsc{since}\ \psi}$	$\rightsquigarrow$	$\boxed{\varphi\ \textsc{since}\ \psi}\boxed{\varphi}$
$\boxed{\varphi\ \textsc{release}\ \psi}$	$\rightsquigarrow$	$\boxed{\varphi, \psi}$	$\boxed{\varphi\ \textsc{release}\ \psi}\Box$	$\rightsquigarrow$	$\boxed{\psi}\boxed{\varphi\ \textsc{release}\ \psi}$
$\boxed{\varphi\ \textsc{initiate}\ \psi}$	$\rightsquigarrow$	$\boxed{\varphi, \psi}$	$\Box\boxed{\varphi\ \textsc{initiate}\ \psi}$	$\rightsquigarrow$	$\boxed{\varphi\ \textsc{initiate}\ \psi}\boxed{\psi}$

3. Replace rules and derivations

An important difference between the strings from the previous section and those considered in model checking (Clarke et al. [4]) is that negated atomic formulas may appear in the former (since atomic formulas are not listed exhaustively). For applications to natural language semantics, it is convenient to build strings from formulas that can be quite complex. A wide class of examples that has to do with so-called continuous change in the progressive is illustrated by *Pat is gaining weight*, for which we might let φ be the formula

$$\exists x\ (\text{weigh(Pat},x) \wedge (\exists y < x)\ \textsc{Previous}\ \text{weigh(Pat},y))$$

and construct the regular language $\boxed{\varphi}^{+}$ for observations $\boxed{\varphi}^{n}$ of $n+1$ moments (for $n \geq 1$). Such applications aside, complex formulas are helpful in incremental analyses of the sets $[\![i, \varphi]\!]$ from the previous section. For instance, the language corresponding to $[\![1, (p \wedge (\top\ \textsc{until}\ q))\ \textsc{until}\ r]\!]$ can be analyzed incrementally as

$$L_1 = \boxed{now}\,\boxed{p, \top\ \textsc{until}\ q}^{*}\,\boxed{r} \tag{7}$$

before unwinding it to a language over the alphabet $\mathsf{Pow}(\{now, p, q, r\})$. It turns out that whereas L_1 is regular (over an alphabet with the symbol $\boxed{p, \top\ \textsc{until}\ q}$ added), the language corresponding to

$$[\![1, (p \wedge (\top\ \textsc{until}\ q))\ \textsc{until}\ r]\!]$$

is *not*. To see this, it is useful to formulate rules of the sort on which a tableau construction for Φ is based. More generally, these rules resemble replace rules in finite-state tool-kits such as that in Beesley and Karttunen [3].

3.1. Replacing Φ-schedules

We shall derive the sets $[\![i, \varphi]\!]$ incrementally by applying certain rules to the Φ-schedule $\{(i, \varphi)\}$. For instance, the top four rule schemas in Table 2 suffice to turn $\{(0, (\textsc{Previous}\ p) \vee \textsc{Next}\ q)\}$ to the set $[\![0, (\textsc{Previous}\ p) \vee \textsc{Next}\ q]\!]$ consisting of the P-schedules $\{(-1, p)\}$ and $\{(1, q)\}$.

The formal definitions are as follows. Given a string s over the alphabet $\mathsf{Pow}(\Phi)$ and an integer $i \in \mathbb{Z}$, let s^{i} be the Φ-strip defined recursively on s by $\epsilon^{i} = \emptyset$ (where ϵ is the empty string) and

$$(\alpha \mathsf{s})^{i} = \{(i, \varphi) \mid \varphi \in \alpha\} \cup \mathsf{s}^{i+1}.$$

Next, given a relation $R \subseteq \mathsf{Pow}(\Phi)^{+} \times \mathsf{Pow}(\Phi)^{+}$ between strings, and Φ-schedules s and s', we say s' *R-succeeds* s if for some $(\mathsf{s}_1, \mathsf{s}_2) \in R$, there exists $i \in \mathbb{Z}$ such that

$$\mathsf{s}_1{}^{i} \subseteq s \text{ and } s' = (s - \mathsf{s}_1{}^{i}) \cup \mathsf{s}_2{}^{i}.$$

(Hence, we can keep s_1 in s' by arranging that s_2 contain s_1.[5]) When R is understood, we shall write $\mathsf{s}_1 \rightsquigarrow \mathsf{s}_2$ and $s \rightsquigarrow s'$ for $(\mathsf{s}_1, \mathsf{s}_2) \in R$ and s' R-succeeds s, respectively. In the example above, we have (for R given by Table 2)

$$\{(0, (\textsc{Previous}\ p) \vee \textsc{Next}\ q)\} \rightsquigarrow \{(0, \textsc{Previous}\ p)\} \rightsquigarrow \{(-1, p)\}$$

and

$$\{(0, (\textsc{Previous}\ p) \vee \textsc{Next}\ q)\} \rightsquigarrow \{(0, \textsc{Next}\ q)\} \rightsquigarrow \{(1, q)\}.$$

Let $\rightsquigarrow^{*}$ be the reflexive transitive closure of $\rightsquigarrow$—i.e. $s \rightsquigarrow^{*} s'$ iff there is a finite sequence $s_1, s_2, \ldots, s_n$ of length $n \geq 1$ such that $s = s_1$, $s' = s_n$ and for $1 \leq m < n$, $s_m \rightsquigarrow s_{m+1}$. Note that if $s \in Str_P$ then there is *no* $i \in \mathbb{Z}$ such that $s(i, \perp)$.

[5] More precisely, $\mathsf{s}_2 \trianglerighteq \mathsf{s}_1$, as defined in Section 4.1.

Proposition 3. *Let $\varphi \in \Phi$ and $i \in \mathbb{Z}$. Then relative to the rules from Table 2,*

$$[\![i,\varphi]\!] \supseteq \{s \in Str_P \mid \{(i,\varphi)\} \rightsquigarrow^* s\}$$

and if neither RELEASE *nor* INITIATE *occur in φ,*

$$[\![i,\varphi]\!] = \{s \in Str_P \mid \{(i,\varphi)\} \rightsquigarrow^* s\}.$$

Before constructing infinite chains of R-successors to analyze temporal formulas built with RELEASE or INITIATE, we need to be careful to restrict applications of the recursive rule schemas

$$\boxed{\varphi \text{ UNTIL } \psi}\,\Box \rightsquigarrow \boxed{\varphi}\boxed{\varphi \text{ UNTIL } \psi} \qquad \Box\,\boxed{\varphi \text{ SINCE } \psi} \rightsquigarrow \boxed{\varphi \text{ SINCE } \psi}\boxed{\varphi}$$

to guarantee the second argument ψ is realized. Accordingly, let R' be the set of rules in Table 2 with these schemas deleted, and

$$\boxed{\varphi \text{ UNTIL } \psi}\,\Box^n \rightsquigarrow \boxed{\varphi}^n\boxed{\psi} \qquad \Box^n\boxed{\varphi \text{ SINCE } \psi} \rightsquigarrow \boxed{\psi}\boxed{\varphi}^n \qquad (n \geq 1)$$

put in their place. An *i-derivation of* φ is a function $f : \mathbb{N} \to Str_{\Phi-\{\bot\}}$ such that $f(0) = \{(i,\varphi)\}$ and conditions (c1) and (c2) hold for all $n \geq 0$.

(c1) $f(n+1)$ R'-succeeds $f(n)$ or $f(n+1) = f(n) \in Str_P$;
(c2) for all $(j,\psi) \in f(n)$ with $\psi \notin P$, there exists $k > n$ such that $(j,\psi) \notin f(k)$.

Condition (c2) ensures for instance, that an i-derivation of

$$(\bot \text{ RELEASE } p) \wedge (\bot \text{ RELEASE } q)$$

attends infinitely often to $\bot$ RELEASE p and $\bot$ RELEASE q alike. Let us write f_P for the function $\lambda n.(f(n) \cap (\mathbb{Z} \times P))$ from $\mathbb{N}$ to Str_P.

Proposition 4. *For all $\varphi \in \Phi$ and $i \in \mathbb{Z}$,*

$$[\![i,\varphi]\!] = \left\{ \bigcup_{n \geq 0} f_P(n) \mid f \text{ is an } i\text{-derivation of } \varphi \right\}$$

and for every i-derivation f of φ and every $n \geq 0$, $f_P(n) \subseteq f_P(n+1)$.

To get a grip on the complexity of the sets $[\![i,\varphi]\!]$, let us, for the sake of definiteness, identify P-strips with strings over the alphabet $\mathrm{Pow}(P \cup \{now\})$ that neither begin nor end with the empty set $\Box$. (For example, we identify $\{(1,p)\}$ with $\boxed{now}\boxed{p}$, rather than any of the strings in $\Box^+\boxed{now}\boxed{p}\Box^* + \boxed{now}\boxed{p}\Box^+$.) Returning to the example of $[\![1, (p \wedge (\top \text{ UNTIL } q)) \text{ UNTIL } r]\!]$ above, observe that q may jump over r when replacing strings in (7),[6] and that the q's never outnumber the p's. That is,

$$[\![1, (p \wedge (\top \text{ UNTIL } q)) \text{ UNTIL } r]\!] \cap \boxed{now}\boxed{p}^+\boxed{r}\boxed{q}^+ = \boxed{now}\sum_{n\geq 1}\sum_{1\leq m\leq n}\boxed{p}^n\boxed{r}\boxed{q}^m$$

which is not regular (by a pumping argument). Hence,

$$[\![1, (p \wedge (\top \text{ UNTIL } q)) \text{ UNTIL } r]\!]$$

[6] The overlap in

$$\boxed{\text{NEXT } q}\boxed{r} \rightsquigarrow \Box\boxed{q,r}$$

reflects the non-atomic structure of sets as symbols (for strips), contrasting with

$$\boxed{\text{NP}}\boxed{\text{VP}} \not\rightsquigarrow \boxed{\text{the}}\boxed{\text{dog,barked}}.$$

In http://www.stanford.edu/~laurik/fsmbook/examples/YaleShooting.html, Karttunen analyzes these sets as strings, advancing the application of methods from Beesley and Karttunen [3] to temporal semantics initiated in [14].

cannot be regular. It does, however, contain the regular language

$$\boxed{now}\left(\boxed{r} + \boxed{p}^{*}\boxed{p,q}\boxed{r} + \boxed{p}^{+}\boxed{q,r} + \boxed{p}^{+}\boxed{r}\square^{*}\boxed{q}\right)$$

that 1-portrays $(p \wedge (\top\ \text{UNTIL}\ q))\ \text{UNTIL}\ r$. We will see in the next section how to extract these regular sublanguages. In the meantime, notice that 0-portrayal yields i-portrayal for any $i \in \mathbb{Z}$, as we can alway prefix φ by i NEXT's for $i > 0$, or $-i$ PREVIOUS's for $i < 0$. Henceforth, let us speak simply of portrayal (understood to mean 0-portrayal).

The sets $[\![i,\varphi]\!]$ are far from unique in satisfying Proposition 2. The empty set portrays every contradiction, as does $\{\emptyset\}$ every tautology φ, whether or not $\emptyset \in [\![0,\varphi]\!]$. An obvious but important fact about portrayal is that if some set of schedules portrays both φ and ψ, then φ and ψ are logically equivalent (in that they are true at the same P-worlds); and conversely, if φ, ψ are logically equivalent, then they are portrayed by the same sets of schedules. Portrayal is determined completely by truth relative to worlds; characterization (of events) is a finer grained notion that is underdetermined by truth at worlds. No world can distinguish p from $p \vee (p \wedge \text{NEXT}\ p)$, but any notion of characterization for event semantics (in linguistics) must discern the difference in temporal extent between p and $p \wedge \text{NEXT}\ p$. Tightening the fit between a formula and a situation it characterizes (over that between a formula and a situation it describes) is essential if, according to Schubert [24], we are to account for causal relations pervasive in natural language.

3.2. Digression: derivations as ways of being true

The replace rules above do not preserve truth $\left(\text{e.g. } \boxed{\varphi \vee \psi} \text{ does not imply } \boxed{\varphi}\right)$, and are in this sense unsound. But they yield ways of explaining truth that when read right to left (rather than left to right) preserve truth $\left(\text{e.g. } \boxed{\varphi} \text{ implies } \boxed{\varphi \vee \psi}\right)$. Moreover, by Propositions 4 and 2(a),

$$v, i \models \varphi \iff (\exists \text{ an } i\text{-derivation } f \text{ of } \varphi)(\forall n \geq 0)\ f_P(n) \subseteq x_v$$

for all functions $v : \mathbb{Z} \to \mathsf{Pow}(P_\circ)$, $i \in \mathbb{Z}$ and $\varphi \in \Phi$. For this reason, it is tempting to describe an i-derivation f of φ as a proof that φ is true at i—assuming that is, f can be grounded in the P-world x_v implicit in talk of truth at i. The reference here to worlds suggests that an i-derivation is not so much a proof as it is *a way of being true*. Before putting proofs aside, however, let us not forget our declared interest in something more than truth at a world; over and above what a formula describes, we want to know what it characterizes (without necessarily bringing in worlds/points). Unraveling what a formula characterizes is a process very much like that of (de)constructing a proof, bottoming out in assumptions that fall well short of a world.

3.3. Digression: observation versus derivation

Let us call a temporal formula $\varphi \in \Phi$ *finitely observable* if $[\![i,\varphi]\!] \subseteq Str_P$ for some (equivalently, any) $i \in \mathbb{Z}$. Examples include temporal formulas with no occurrence of RELEASE or INITIATE (Proposition 2(b)). These are also finitely derivable by Proposition 3, raising the question: can we sharpen Proposition 3 to

$$[\![i,\varphi]\!] \cap Str_P = \{s \in Str_P \mid \{(i,\varphi)\} \rightsquigarrow^{*} s\}\ ?$$

No. Consider $\perp\ \text{RELEASE}\ \top$. As $\emptyset \in [\![j,\top]\!]$ for all $j \geq i$, $\lambda n.\emptyset$ belongs to $F_{+}(\top, i)$ and so $\emptyset \in [\![i, \perp\ \text{RELEASE}\ \top]\!]$. On the other hand, there is no $s \in Str_P$ such that $\{(i, \perp\ \text{RELEASE}\ \top)\} \rightsquigarrow^{*} s$. Instead, $\{(i, \perp\ \text{RELEASE}\ \top)\} \rightsquigarrow \{(i,\perp),(i,\top)\}$ and $\{(i, \perp\ \text{RELEASE}\ \top)\} \rightsquigarrow \{(i,\top),(i+1, \perp\ \text{RELEASE}\ \top)\}$. In sum, the formula $\perp\ \text{RELEASE}\ \top$ is finitely observable but *not* finitely derivable

$$\begin{aligned} [\![i, \perp\ \text{RELEASE}\ \top]\!] &= \{\emptyset\} \\ &\neq \{s \in Str_P \mid \{(i, \perp\ \text{RELEASE}\ \top)\} \rightsquigarrow^{*} s\} \ = \ \emptyset. \end{aligned}$$

The inequality above rests on conceptualizing an i-derivation $f : \mathbb{N} \to Str_\Phi$ as a computation that has at the nth stage $f(n)$ processed the approximation $f_P(n) = f(n) \cap (\mathbb{Z} \times P)$. There is a striking similarity here to notions of stage and algorithm investigated by Moschovakis; in particular, readers familiar with Moschovakis [19] might draw the following analogies with Fregean sense (*Sinn*) and denotation (*Bedeutung*).

$$\frac{\text{observation}}{\text{derivation}} \approx \frac{\text{value}}{\text{computation}} \approx \frac{\text{denotation}}{\text{sense}}$$

As the mode or manner of presenting denotation (value), sense (computation) can be a very delicate matter, less robust than denotation (value). I would not claim that a definitive account of sense for formulas in Φ necessarily exists, let alone that the notion of derivation above provides such an account. The preceding discussion of $\perp\ \text{RELEASE}\ \top$ brings out a problem with waiting for the end of an i-derivation before "observing" the value under construction. The discussion also points to a possible solution: step up to higher types and form the lambda term $\lambda n.\emptyset$ in $F_{+}(\top, i)$. Type theory construed proof-thoeretically has in recent years been applied fruitfully to interpret (and not only assemble) semantic representations for natural language (e.g. Ranta [22] and Cooper [5]). But a worry noted by Sundholm (among others) some 20 years ago has persisted:

> . . . it is not at all clear that one can export the 'canonical proof-objects' conception of meaning outside the confined area of constructive mathematics. In particular, the treatment of atomic sentences such as OWN[x, y] is left intolerably vague . . . and it is an open problem how to remove that vagueness ([25], p. 503).

T. Parsons has dubbed "the study of those 'formulas of English' that are treated as atomic formulas in most logical investigations of English" *subatomic semantics* ([20], p. ix), hypothesizing that events are the key to its mysteries. Assuming an event can be formulated as a string (or, more precisely a P-strip, for some set P of observations), it is natural to try out automata-theoretic methods (firmly established in model checking as well as various areas of natural language processing such as morphology) before moving on to the more complex type-theoretic machinery of proof-conditional semantics. Insofar as events in subatomic semantics can always be put in the past, the strings that represent them must arguably be finite. The complications infinite strings pose are substantial enough for us to keep infinite strings out of subatomic semantics, if we can. For instance, $[\![1, \bot \text{ RELEASE } (p \vee q)]\!]$ is uncountable (for $p \neq q$), although it is trivial to mechanically generate all its finite approximations $\boxed{now}\left(\boxed{p} + \boxed{q}\right)^+$.

An instructive English example exposing the shortcomings in reducing p BEFORE q to q AFTER p is

(∗) Pat stopped the car before it hit the tree.

A postcondition of Pat stopping the car is that car be stationary. The car can be assumed to remain stationary unless some force puts it in motion. As being in motion is a precondition for hitting the tree, we may conclude from (∗) that in the absence of any intervening forces, the car did not hit the tree. To formalize such reasoning, it is convenient to pick out certain formulas as *inertial*, and for each inertial φ, build a non-inertial formula fφ saying intuitively that a force is applied to make φ true at the next moment ([10]). Assuming the negation of an inertial formula is inertial, we can then equate an inertial φ with the conjunction

$$(\mathsf{f}\overline{\varphi} \text{ RELEASE } \varphi) \wedge \text{PREVIOUS } (\mathsf{f}\varphi \text{ INITIATE NEXT } \varphi)$$

which says not only that φ holds, but that it persists forward unless some force is applied against it, and persists backward unless some force was applied for it.[7] We can encode inertial persistence either through the replace rule

$$\Box\boxed{\varphi} \rightsquigarrow \boxed{\mathsf{f}\varphi \text{ INITIATE NEXT } \varphi}\boxed{\mathsf{f}\overline{\varphi} \text{ RELEASE } \varphi}$$

or, avoiding explicit reference to either RELEASE or INITIATE, through constraints

$$\boxed{\varphi}\Box \Rightarrow \Box\boxed{\varphi} + \boxed{\mathsf{f}\overline{\varphi}}\Box \tag{8}$$

$$\Box\boxed{\varphi} \Rightarrow \boxed{\varphi}\Box + \boxed{\mathsf{f}\varphi}\Box \tag{9}$$

that define regular languages according to a modification (from [14]) of constructs in Beesley and Karttunen [3]. Precisely what $\Rightarrow$ means, and how $\Rightarrow$ can be applied to overcome the non-regularity above are what the next section is all about.

4. Constraints as languages

In this section, we recast the replace rules from the preceding section as constraints defined through a subsumption relation $\trianglerighteq$ on languages that compares the information encoded in strings of the same length. The pay-off is that the sets $[\![i, \varphi]\!]$ from Section 2 are reduced to regular languages that cover (in a precise sense) not only the irregular language corresponding to $[\![1, (p \wedge (\top \text{ UNTIL } q)) \text{ UNTIL } r]\!]$ but also infinite strings from INITIATE and RELEASE.

Throughout this section, we work with the alphabet $\mathsf{Pow}(\Phi \cup \{now\})$, forming the set now_Φ of *now-anchored* strings where *now* occurs exactly once

$$\begin{aligned} now_\Phi &= \{\alpha_1 \cdots \alpha_n \in \mathsf{Pow}(\Phi \cup \{now\})^+ \mid \exists \text{ exactly one } i \text{ such that} \\ &\qquad\qquad 1 \leq i \leq n \text{ and } now \in \alpha_i\} \\ &= \mathsf{Pow}(\Phi)^* \{\alpha \cup \boxed{now} \mid \alpha \subseteq \Phi\} \mathsf{Pow}(\Phi)^*. \end{aligned}$$

[7] Differentiating fφ from f$\overline{\varphi}$ allows us to equate fφ with

$$\mathsf{f}\varphi \wedge (\mathsf{f}\overline{\varphi} \vee \text{NEXT } \varphi)$$

asserting that fφ succeeds in bringing φ about at the next moment unless opposed.

Using the semantics for $\Rightarrow$ and $L^{\trianglerighteq}$ given below, we can capture now_{Φ} by applying the constraint

$$\boxed{now}\Box^{*}\boxed{now} \Rightarrow \emptyset \tag{10}$$

precluding multiple occurrences of *now* to the language

$$\left(\Box^{*}\boxed{now}\Box^{*}\right)^{\trianglerighteq}$$

asserting that *now* occurs.

4.1. Subsumption and constraints

Given strings s and s′ over the alphabet $\mathsf{Pow}(\Phi \cup \{now\})$, we say s *subsumes* s′ and write $s \trianglerighteq s'$ if s and s′ have the same length and s componentwise includes s′

$$\alpha_1\alpha_2\cdots\alpha_n \trianglerighteq \beta_1\beta_2\cdots\beta_m \quad \text{iff} \quad n = m \text{ and } \alpha_i \supseteq \beta_i \text{ for } 1 \leq i \leq n.$$

For languages L and $L' \subseteq \mathsf{Pow}(\Phi \cup \{now\})^{*}$, L *subsumes* L' if each string in L subsumes some string in L'

$$L \trianglerighteq L' \quad \text{iff} \quad (\forall s \in L)(\exists s' \in L')\ s \trianglerighteq s'.$$

These definitions allow us to conflate a string s with the singleton language {s}. As a type with instances $s \in L$, a language L is essentially a disjunction $\bigvee_{s \in L}$ s of conjunctions s; between strings, more is more

$$\text{e.g.} \quad \boxed{\varphi, \psi} \trianglerighteq \boxed{\varphi}$$

but between languages, less is more

$$\text{e.g.} \quad \boxed{\varphi} \trianglerighteq \boxed{\varphi} + \boxed{\psi}.$$

A *now*-anchored string s encodes the schedule $sched(s) \subseteq \mathbb{Z} \times \Phi$ with *now* taken as time 0 and

$$(i, \varphi) \in sched(s) \iff s \trianglerighteq \Box^{*} stg(\varphi, i) \Box^{*}$$

where

$$stg(\varphi, i) = \begin{cases} \boxed{now, \varphi} & \text{if } i = 0 \\ \boxed{now}\Box^{i-1}\boxed{\varphi} & \text{if } i > 0 \\ \boxed{\varphi}\Box^{1-i}\boxed{now} & \text{if } i < 0. \end{cases}$$

For example,

$$sched\left(\boxed{p}\Box\boxed{now}\boxed{q \wedge p, r}\right) = \{(-2, p), (1, q \wedge p), (1, r)\}.$$

For an arbitrary *now*-anchored string $s = \alpha_1 \cdots \alpha_n$ with $now \in a_i$, we can re-express $sched(s)$ using the notation from Section 3.1 as

$$sched(s) = s_{\circ}{}^{1-i}$$

where $s_{\circ}$ is s without *now*.

To reformulate replace rules from the preceding section in terms of regular languages, we define the *constraint* $L \Rightarrow L'$ (for any languages L and $L' \subseteq \mathsf{Pow}(\Phi \cup \{now\})^{*}$) to be the set of strings $s \in \mathsf{Pow}(\Phi \cup \{now\})^{*}$ such that $s \trianglerighteq \Box^{n} L' \Box^{m}$ whenever $s \trianglerighteq \Box^{n} L \Box^{m}$

$$s \in L \Rightarrow L' \iff (\forall n, m \geq 0)\ s \trianglerighteq \Box^{n} L \Box^{m} \text{ implies } s \trianglerighteq \Box^{n} L' \Box^{m}.$$

If we collect the strings that subsume L in $L^{\trianglerighteq}$

$$L^{\trianglerighteq} = \{s \in \mathsf{Pow}(\Phi \cup \{now\})^{*} \mid s \trianglerighteq L\}$$

and write $\overline{L}$ for the set-theoretic complement $\mathsf{Pow}(\Phi \cup \{now\})^{*} - L$, then

$$L \Rightarrow L' = \overline{\mathsf{Pow}(\Phi \cup \{now\})^{*}\ (L^{\trianglerighteq} \cap \overline{(L')^{\trianglerighteq}})\ \mathsf{Pow}(\Phi \cup \{now\})^{*}}$$

Table 3
Constraints for $\varphi, \psi \in \Phi$.

$$\boxed{\varphi \wedge \psi} \Rightarrow \boxed{\varphi, \psi}$$

$$\boxed{\varphi \vee \psi} \Rightarrow \boxed{\varphi} + \boxed{\psi}$$

$$\boxed{\text{Next } \varphi}\square \Rightarrow \square\boxed{\varphi} \tag{11}$$

$$\square\boxed{\text{Previous } \varphi} \Rightarrow \boxed{\varphi}\square \tag{12}$$

$$\boxed{\varphi \text{ UNTIL } \psi}\square \Rightarrow \boxed{\psi}\square + \boxed{\varphi}\boxed{\varphi \text{ UNTIL } \psi} \tag{13}$$

$$\square\boxed{\varphi \text{ SINCE } \psi} \Rightarrow \square\boxed{\psi} + \boxed{\varphi \text{ SINCE } \psi}\boxed{\varphi} \tag{14}$$

$$\boxed{\varphi \text{ RELEASE } \psi}\square \Rightarrow \boxed{\varphi, \psi}\square + \boxed{\psi}\boxed{\varphi \text{ RELEASE } \psi}$$

$$\square\boxed{\varphi \text{ INITIATE } \psi} \Rightarrow \square\boxed{\varphi, \psi} + \boxed{\varphi \text{ INITIATE } \psi}\boxed{\psi}$$

([3,14]). It follows that if L and L' are regular languages, so is $L \Rightarrow L'$. For $L' = \emptyset$ as in (10), $L \Rightarrow \emptyset$ reduces to the language of strings that do not subsume $\square^* L \square^*$

$$L \Rightarrow \emptyset = \overline{\left(\square^* L \square^*\right)^{\trianglerighteq}}.$$

The replace rules in Table 2 suggest the constraints in Table 3, with, for instance, (13) picking out strings $\alpha_1 \cdots \alpha_n$ such that for all $i < n$,

$$\text{‘}\varphi \text{ UNTIL } \psi\text{’} \in \alpha_i \text{ implies } \psi \in \alpha_i \text{ or } (\varphi \in \alpha_i \text{ and ‘}\varphi \text{ UNTIL } \psi\text{’} \in \alpha_{i+1}).$$

To apply the constraints in Table 3, some definitions are helpful. Let $L_{\trianglerighteq}$ consist of the strings in L that are $\trianglerighteq$-minimal

$$L_{\trianglerighteq} = \{s \in L \mid \text{not } (\exists s' \in L - \{s\})\ s \trianglerighteq s'\}$$

and let $\equiv$ be the equivalence on languages induced by $\trianglerighteq$

$$L \equiv L' \iff L \trianglerighteq L' \text{ and } L' \trianglerighteq L.$$

Observe that

$$L_{\trianglerighteq} \ \equiv \ L \ \equiv \ L^{\trianglerighteq}$$

and that whenever $L \equiv L'$,

$$L_{\trianglerighteq} \ \subseteq \ L' \ \subseteq \ L^{\trianglerighteq}.$$

Next, we define the *application of C to L* by

$$apply(C, L) = (L^{\trianglerighteq} \cap C)_{\trianglerighteq}$$

([10]) so that for example,

$$apply\left(\boxed{p \wedge q} \Rightarrow \boxed{p, q}, \boxed{p \wedge q}\right) = \boxed{p, q, p \wedge q}.$$

For C_0 equal to the constraint

$$\boxed{p \text{ UNTIL } q}\square \Rightarrow \left(\boxed{q}\square + \boxed{p}\boxed{p \text{ UNTIL } q}\right)$$

we get

$$apply\left(C_0, \boxed{p \text{ UNTIL } q}\square^*\right) = \boxed{p \text{ UNTIL } q,\ p}^*\boxed{p \text{ UNTIL } q,\ q}\square^+ + \boxed{p \text{ UNTIL } q,\ p}^*\boxed{p \text{ UNTIL } q}$$

where the latter disjunct $\boxed{p \text{ UNTIL } q,\ p}^*\boxed{p \text{ UNTIL } q}$ fails to require that q eventually hold. To plug this loophole, let us replace the UNTIL-constraint (13) in Table 3 by

$$\boxed{\varphi \text{ UNTIL } \psi} \Rightarrow \boxed{\psi} + \boxed{\varphi,\ \varphi \text{ UL } \psi} \tag{15}$$

$$\boxed{\varphi \text{ UL } \psi} \overset{a}{\Rightarrow} \boxed{\varphi}^{*}\boxed{\psi} \tag{16}$$

where $L \overset{a}{\Rightarrow} L'$ is read "if L then *a*fterwards L'"

$$L \overset{a}{\Rightarrow} L' = \overline{\left(\Box^{*}L\right)^{\trianglerighteq}\ \overline{(L'\Box^{*})^{\trianglerighteq}}}.$$

If C_0' is (15) $\cap$ (16) for $\varphi = p$ and $\psi = q$ then

$$apply\left(C_0',\ \boxed{p \text{ UNTIL } q}\Box^{*}\right) = \boxed{p \text{ UNTIL } q,\ q}\Box^{*} + \boxed{p \text{ UNTIL } q,\ p \text{ UL } q}\boxed{p}^{*}\boxed{q}\Box^{*}.$$

Similarly, we replace the SINCE-constraint (14) in Table 3 by

$$\boxed{\varphi \text{ SINCE } \psi} \Rightarrow \boxed{\psi} + \boxed{\varphi,\ \varphi \text{ SI } \psi} \tag{17}$$

$$\boxed{\varphi \text{ SI } \psi} \overset{b}{\Rightarrow} \boxed{\psi}\boxed{\varphi}^{*} \tag{18}$$

where $L \overset{b}{\Rightarrow} L'$ is read "if L then *b*eforehand L'"

$$L \overset{b}{\Rightarrow} L' = \overline{\overline{\left(\Box^{*}L'\right)^{\trianglerighteq}}\ (L\Box^{*})^{\trianglerighteq}}.$$

We can also strengthen the NEXT-constraint (11) to

$$\boxed{\text{NEXT } \varphi} \overset{a}{\Rightarrow} \boxed{\varphi} \tag{19}$$

and the PREVIOUS-constraint (12) to

$$\boxed{\text{PREVIOUS } \varphi} \overset{b}{\Rightarrow} \boxed{\varphi}. \tag{20}$$

The variants $\overset{a}{\Rightarrow}$ and $\overset{b}{\Rightarrow}$ of $\Rightarrow$ build in eventuality conditions, sidestepping complications from compactness arguments (familiar from first-order logic) deriving $\boxed{p}^{\infty}$ from $\boxed{p}^{+}$. We pay particular attention next to the formation of infinite strings.

4.2. Padding and paths

Given *now*-anchored strings s and s$'$, we say s *weakly subsumes* s$'$ and write s$\underline{\blacktriangleright}s'$ when $sched(\text{s}')$ is a subset of $sched(\text{s})$

$$\text{s}\underline{\blacktriangleright}\text{s}' \iff sched(\text{s}') \subseteq sched(\text{s}).$$

We can reduce $\underline{\blacktriangleright}$ to $\trianglerighteq$ and relax the assumption that s and s$'$ are *now*-anchored by defining a function *unpad* that removes all initial and final empty boxes $\Box$ in a string

$$unpad(\text{s}) = \begin{cases} \text{s} & \text{if s neither begins nor ends with } \Box \\ unpad(\text{s}') & \text{if s} = \Box\text{s}' \text{ or else if s} = \text{s}'\Box \end{cases}$$

and by agreeing that s$\underline{\blacktriangleright}s'$ iff s subsumes a string that is *unpad*-equivalent with s$'$

$$\text{s}\underline{\blacktriangleright}\text{s}' \quad \text{iff} \quad (\exists \text{s}'')\ unpad(\text{s}'') = unpad(\text{s}') \text{ and } \text{s} \trianglerighteq \text{s}''.$$

Stepping from strings up to languages, let us write $unpad(L)$ for the image of L under *unpad*

$$unpad(L) = \{unpad(\text{s}) \mid \text{s} \in L\}$$

and $L^{\Box}$ for L with any number of leading and trailing $\Box$'s deleted or added

$$L^{\Box} = \Box^{*}unpad(L)\Box^{*}$$

$$= \{\mathsf{s} \in \mathsf{Pow}(\Phi \cup \{now\})^* \mid unpad(\mathsf{s}) \in unpad(L)\}.$$

The equivalence

$$L \underline{\blacktriangleright} L' \iff (\forall \mathsf{s} \in L)(\exists \mathsf{s}' \in L')\ \mathsf{s} \underline{\blacktriangleright} \mathsf{s}'$$

(paralleling $\trianglerighteq$) is a consequence of defining $\underline{\blacktriangleright}$ from $\trianglerighteq$ by weakening the second argument L' to $L'^{\Box}$

$$L \underline{\blacktriangleright} L' \iff L \trianglerighteq L'^{\Box}.$$

Given a schedule $x \subseteq \mathbb{Z} \times \Phi$, we can form approximations of x along a set L of *now*-anchored strings, defining

$$Sched(L) = \bigcup_{\mathsf{s} \in L} sched(\mathsf{s}).$$

Of particular interest below are sets L that are similar to the set $Stg(x)$ of *now*-anchored strings s inducing schedules contained in x

$$Stg(x) = \{\mathsf{s} \in now_{\Phi} \mid sched(\mathsf{s}) \subseteq x\}.$$

A *path* is a set L of *now*-anchored strings that is

(i) $\underline{\blacktriangleright}$*-directed* in that for all $\mathsf{s}, \mathsf{s}' \in L$, there exists $\mathsf{s}'' \in L$ such that

$$\mathsf{s}'' \underline{\blacktriangleright} \mathsf{s} \text{ and } \mathsf{s}'' \underline{\blacktriangleright} \mathsf{s}'$$

and

(ii) *extensible* in that for every $\mathsf{s} \in L$, there exists $\mathsf{s}' \in L$ such that

$$\mathsf{s}' \trianglerighteq \Box^+ \mathsf{s} \Box^+.$$

Note that for all $x \subseteq \mathbb{Z} \times \Phi$, $Stg(x)$ is a path.

Let us call L *stripped* if $Sched(L)$ is finite (i.e. $Sched(L) \in Str_{\Phi \cup \{now\}}$), and say that a string s *carves out* L if $sched(s) = Sched(L)$. For example, a *now*-anchored string s carves out the path $\Box^* \mathsf{s} \Box^*$. Clearly, if L is a path, then

$$\begin{aligned} L \text{ is stripped } &\iff \text{some string in } L \text{ carves out } L \\ &\iff (\exists \mathsf{s} \in L)(\forall \mathsf{s}' \in L)\ \mathsf{s} \underline{\blacktriangleright} \mathsf{s}'c \end{aligned}$$

and

$$\mathsf{s} \text{ carves out } L \iff (\forall n > 0)(\exists l > n)(\exists m > n)\ \Box^l \mathsf{s} \Box^m \in L$$

(with quantification similar to acceptance in Büchi automata; e.g. Clarke et al. [4]).

4.3. Approximating $[\![i, \varphi]\!]$

Let $\mathcal{C}$ consist of the constraints from Table 3 with (11)–(14) replaced by (15)–(20). Henceforth, we assume that Φ is closed also under the binary connectives UL and SI, with

$$\begin{aligned} [\![i, \varphi \text{ UL } \psi]\!] &= [\![i, \varphi \text{ UNTIL } \psi]\!] - [\![i, \psi]\!] \\ [\![i, \varphi \text{ SI } \psi]\!] &= [\![i, \varphi \text{ SINCE } \psi]\!] - [\![i, \psi]\!]. \end{aligned}$$

For every $\varphi \in \Phi$, let $\mathcal{C}_{\varphi}$ be the intersection of constraints $L_1 \Rightarrow L_2$ in $\mathcal{C}$ such that the lefthand side L_1 contains only subformulas of φ (suitably modified for UNTIL and SINCE)

$$\mathcal{C}_{\varphi} = \bigcap \{(L_1 \Rightarrow L_2) \in \mathcal{C} \mid L_1 \subseteq \mathsf{Pow}(subfmla(\varphi))^*\}$$

where for $p \in P$,

$$subfmla(p) = \{p\},$$

for $U \in \{\textsc{Next}, \textsc{Previous}\}$,

$$subfmla(U\varphi) = \{U\varphi\} \cup subfmla(\varphi)\ ,$$

for $B \in \{\vee, \wedge, \text{RELEASE}, \text{INITIATE}, \text{UL}, \text{SI}\}$,

$$subfmla(\varphi B \psi) = \{\varphi B \psi\} \cup subfmla(\varphi) \cup subfmla(\psi)$$

and

$$subfmla(\varphi \text{ UNTIL } \psi) = \{\varphi \text{ UNTIL } \psi,\ \varphi \text{ UL } \psi\} \cup subfmla(\varphi) \cup subfmla(\psi)$$
$$subfmla(\varphi \text{ SINCE } \psi) = \{\varphi \text{ SINCE } \psi,\ \varphi \text{ SI } \psi\} \cup subfmla(\varphi) \cup subfmla(\psi).$$

For every $\varphi \in \Phi$, $subfmla(\varphi)$ is finite, and so $\mathcal{C}_\varphi$ is regular. For example, if φ is $\textsc{Previous}(p) \wedge \textsc{Next}(q)$, $\mathcal{C}_\varphi$ is

$$\left(\boxed{\textsc{Previous}(p) \wedge \textsc{Next}(q)} \Rightarrow \boxed{\textsc{Previous}(p), \textsc{Next}(q)}\right)$$
$$\cap \left(\Box\boxed{\textsc{Previous}(p)} \Rightarrow \boxed{p}\Box\right) \cap \left(\boxed{\textsc{Next}(q)}\Box \Rightarrow \Box\boxed{q}\right)$$

and $apply\left(\mathcal{C}_\varphi,\ \Box\boxed{\textsc{Previous}(p) \wedge \textsc{Next}(q)}\Box\right)$ consists of the single string

$$\hat{\mathsf{s}} = \boxed{p}\boxed{now, \textsc{Previous}(p), \textsc{Next}(q), \textsc{Previous}(p) \wedge \textsc{Next}(q)}\boxed{q}.$$

To restrict a string s to $P_{now} = P \cup \{now\}$, we form $\mathsf{r}_P(\mathsf{s})$ by intersection with P_{now}

$$\mathsf{r}_P(\alpha_1 \cdots \alpha_n) = (\alpha_1 \cap P_{now}) \cdots (\alpha_n \cap P_{now})$$

so that for example, $\mathsf{r}_P(\hat{\mathsf{s}}) = \boxed{p}\boxed{now}\boxed{q}$.

Proposition 5. *For every $\varphi \in \Phi$ constructed from P using $\wedge$ and at most n applications of* $\textsc{Next}$ *and at most n applications of* $\textsc{Previous}$,

$$[\![0, \varphi]\!] = \left\{sched(\mathsf{r}_P(\mathsf{s})) \mid \mathsf{s} \in apply\left(\mathcal{C}_\varphi,\ \Box^n\boxed{now, \varphi}\Box^n\right)\right\}.$$

Proposition 5 fails for φ built with $\vee$—take, for example, $p \vee (p \wedge \textsc{Next}(p))$—but holds with = replaced by $\supseteq$. Also, to avoid the pesky parameter n in Proposition 5, we apply $\mathcal{C}_\varphi$ to $\Box^*\boxed{now, \varphi}\Box^*$, obtaining in the case of $\varphi = \textsc{Previous}(p) \wedge \textsc{Next}(q)$, $\left(\epsilon + \Box^*\boxed{p}\right)\mathsf{s}\left(\epsilon + \boxed{q}\Box^*\right)$ where s is

$$\boxed{now, \textsc{Previous}(p), \textsc{Next}(q), \textsc{Previous}(p) \wedge \textsc{Next}(q)}.$$

Restricting to P_{now}, we let $\mathsf{r}_P(L) = \{\mathsf{r}_P(\mathsf{s}) \mid \mathsf{s} \in L\}$ and define

$$\mathcal{C}_{now}(\varphi) = \mathsf{r}_P\left(apply\left(\mathcal{C}_\varphi,\ \Box^*\boxed{now, \varphi}\Box^*\right)\right)$$

so that

$$\mathcal{C}_{now}(\textsc{Previous}(p) \wedge \textsc{Next}(q)) = \left(\epsilon + \Box^*\boxed{p}\right)\boxed{now}\left(\epsilon + \boxed{q}\Box^*\right)$$

and

$$\mathcal{C}_{now}(p \text{ UNTIL } q) = \Box^*\left(\boxed{now, q} + \boxed{now, p}\boxed{p}^*\boxed{q}\right)\Box^*.$$

Notice that padding $\boxed{now, \varphi}$ in $\mathcal{C}_{now}(\varphi)$ is essential for eking out an extensible set from $\mathcal{C}_{now}(\varphi)$. Finally, for every $\varphi \in \Phi$, we form the set

$$\mathcal{S}(\varphi) = \{Sched(L) \mid L \subseteq \mathcal{C}_{now}(\varphi) \text{ and } L \text{ is a path}\}$$

of schedules given by paths within $\mathcal{C}_{now}(\varphi)$.

Proposition 6. *For every $\varphi \in \Phi$, $\mathcal{S}(\varphi) \subseteq [\![0, \varphi]\!]$. Conversely, for all $x \in [\![0, \varphi]\!]$, there exists $y \in \mathcal{S}(\varphi)$ such that $y \subseteq x$.*

For $n > 0$, $[\![n, \varphi]\!]$ and $[\![-n, \varphi]\!]$ are reducible to

$$[\![0, \textsc{Next}^n(\varphi)]\!] \text{ and } [\![0, \textsc{Previous}^n(\varphi)]\!]$$

respectively. Proposition 6 is sufficient for $\mathcal{S}(\varphi)$ to induce the same notion of description via (1). For φ built with neither RELEASE nor INITIATE, Proposition 2(b) says $[\![0, \varphi]\!] \subseteq Str_P$ whence

$$\mathcal{S}(\varphi) = \{sched(\mathsf{s}) \mid \mathsf{s} \in \mathcal{C}_{now}(\varphi)\}.$$

(It is crucial here that (11) and (12) are strengthened to (19) and (20).) For every $\varphi \in \Phi$, one can build finite-state machines accepting $\mathcal{C}_{now}(\varphi)$. For $\hat{\varphi} = (p \wedge (\top \text{ UNTIL } q)) \text{ UNTIL } r$ from *Section* 3, we have $unpad(\mathcal{C}_{now}(\textsc{Next}\hat{\varphi})) = \boxed{now}\,\hat{L}$ where

$$\hat{L} = \boxed{r} + \boxed{p}^{*}\boxed{p,q}\boxed{r} + \boxed{p}^{+}\boxed{q,r} + \boxed{p}^{+}\boxed{r}\square^{*}\boxed{q}$$

with at most one occurrence of q in every $\hat{L}$-string because of $\trianglerighteq$-minimization (from *apply*).

5. Discussion

Summing up, Section 2 proposed definitions $[\![i, \varphi]\!]$ of the set of situations that a formula φ of Linear Temporal Logic at time i characterizes; Section 3 derived $[\![i, \varphi]\!]$ from (i, φ) by replace rules; and Section 4 applied declarative (order-independent) constraints (instead of order-sensitive procedures) to extract subsets of $[\![i, \varphi]\!]$ that define the same notion of description (under (1)). The denotational semantics provided by the constraints in Section 4 admit a simpler (finite-state) implementation than the operational semantics of Section 3.

The focus above on characterizing (as opposed to describing) situations reflects a shift in emphasis from models to formulas. Whereas the choice of a model (in the form of a machine or program) can be taken for granted in model checking, it is often unclear in natural language applications what models to consider. Whatever these models may be, however, do formulas that characterize situations as strings presuppose models where time is discrete? Must we throw out the real line $\Re$ as a model of time? No, no. Situations-as-strings presupposes only that we can make some selection of discrete times—e.g. $\mathbb{Z}$ from $\Re$. Any such selection can be refined indefinitely. Indeed, we can extract $\Re$ through inverse limits relative to projections respecting (a1) and (a2).

(a1) every string is formed from a finite alphabet (i.e. finitely many observations);
(a2) time does not advance without change.

More precisely, fix a set Ψ of formulas that includes copies of all real numbers. Keeping (a1) in mind, let $Fin(\Psi)$ be the set of all finite subsets of Ψ. We build (a2) into a function $\rho : \mathsf{Pow}(\Psi)^* \to \mathsf{Pow}(\Psi)^*$ that reduces a block $\alpha\alpha$ of two α's to one

$$\rho(\mathsf{s}) = \begin{cases} \mathsf{s} & \text{if length}(\mathsf{s}) \leq 1 \\ \rho(\alpha\mathsf{s}') & \text{if } \mathsf{s} = \alpha\alpha\mathsf{s}' \\ \alpha\rho(\alpha'\mathsf{s}') & \text{if } \mathsf{s} = \alpha\alpha'\mathsf{s}' \text{ with } \alpha \neq \alpha' \end{cases}$$

for all sets $\alpha, \alpha' \subseteq \Psi$. For example, $\rho\left(\square\square\boxed{p}\square\boxed{p}\boxed{p}\right) = \square\boxed{p}\square\boxed{p}$. For every $\Psi' \in Fin(\Psi)$, let $\rho_{\Psi'} : \mathsf{Pow}(\Psi)^* \to \mathsf{Pow}(\Psi')^*$ be ρ composed with $\mathsf{r}_{\Psi'}$

$$\rho_{\Psi'}(\mathsf{s}) = \rho(\mathsf{r}_{\Psi'}(\mathsf{s}))$$

where $\mathsf{r}_{\Psi'}$ componentwise intersects strings with Ψ'

$$\mathsf{r}_{\Psi'}(\alpha_1 \cdots \alpha_n) = (\alpha_1 \cap \Psi') \cdots (\alpha_n \cap \Psi')$$

for all $\alpha_1 \cdots \alpha_n \in \mathsf{Pow}(\Psi)^*$. For example, given real numbers $r_1 < r_2 < r_3$,

$$\begin{aligned} \rho_{\{r_1,r_3\}}\left(\square\boxed{r_1}\square\boxed{r_2}\square\boxed{r_3}\square\right) &= \rho\left(\square\boxed{r_1}\square\square\square\boxed{r_3}\square\right) \\ &= \square\boxed{r_1}\square\boxed{r_3}\square. \end{aligned}$$

The inverse limit of the projections $(\rho_{\Psi'})_{\Psi' \in Fin(\Psi)}$

$$\mathbf{IL}(\Psi) = \left\{ \sigma \in \prod_{\Psi' \in Fin(\Psi)} \mathsf{Pow}(\Psi')^* \;\middle|\; \right.$$

$$\left.\sigma(\Psi'') = \rho_{\Psi''}(\sigma(\Psi')) \text{ for all } \Psi'' \subseteq \Psi' \in Fin(\Psi)\right\}$$

includes points σ such that for every finite set $\Psi' = \{r_1, r_2, \ldots, r_n\} \subseteq \Re$ of reals with $r_1 < r_2 < \cdots < r_n$,

$$\sigma(\Psi') = \boxed{}\boxed{r_1}\boxed{}\boxed{r_2}\boxed{}\cdots\boxed{r_n}\boxed{}.$$

In general, each $\sigma \in \mathbf{IL}(\Psi)$ defines a strict partial order $\prec_\sigma$ on

$$\left\{\psi \in \Psi \mid \sigma(\{\psi\}) = \boxed{}\boxed{\psi}\boxed{}\right\}$$

such that for all $\psi, \psi' \in \Psi$,

$$\psi \prec_\sigma \psi' \iff \sigma(\{\psi, \psi'\}) = \boxed{}\boxed{\psi}\boxed{}\boxed{\psi'}\boxed{}.$$

Let us call a string s ρ*-reduced* if $\rho(\mathsf{s}) = \mathsf{s}$. Note that for all $\sigma \in \mathbf{IL}(\Psi)$ and $\Psi' \in Fin(\Psi)$, $\sigma(\Psi')$ is ρ-reduced. How then do we justify the use of strings s for $[\![i, \varphi]\!]$ that are *not* ρ-reduced? By appealing to the boundedness of observations; the observables in Ψ' exist alongside the un-observables in $\Psi - \Psi'$. For any language L over Pow(Ψ'), it is enough that for some formula tick $\in \Psi - \Psi'$, we apply the intersection of the constraints

$$\boxed{}\boxed{} \Rightarrow \boxed{\text{tick}}\boxed{} + \boxed{}\boxed{\text{tick}}$$

$$\boxed{\text{tick}}\boxed{\text{tick}} \Rightarrow \emptyset$$

to L to obtain strings that are ρ-reduced. Or we might refine the tick's to countably many fresh formulas $\{\chi_i\}_{i\in\mathbb{Z}}$ appearing in the constraints

$$\boxed{\chi_i}\boxed{} \Rightarrow \boxed{}\boxed{\chi_{i+1}} \tag{21}$$

$$\boxed{}\boxed{\chi_i} \Rightarrow \boxed{\chi_{i-1}}\boxed{} \tag{22}$$

$$\boxed{\chi_i}\boxed{}^*\boxed{\chi_i} \Rightarrow \emptyset \tag{23}$$

tagging strings with successor chains on which the connectives PREVIOUS and NEXT from LTL depend. As $\mathbb{Z}$ is infinite, however, the infinitely many constraints (21)–(23) lead to an infinite alphabet and a non-regular language. Accordingly, we have put ρ aside in Sections 1–4, and made do with χ_0, called *now*.

That said, constraints (21) and (22) raise the question: are there formulas in Ψ (or Φ) which can be viewed as atomic in that they never appear in the lefthand side L of constraints $L \Rightarrow L'$ with $L' \neq \emptyset$, or in the lefthand side of replace rules such as those in Table (2)? Under Table (2), derivations terminate as soon as all boxes in a string contain only propositions in P; that is, propositions in P form terminal symbols, and formulas in $\Phi - P$ non-terminals (so that strings over subsets of P are in normal form). The inertial constraints (8) and (9) mentioned at the end of Section 3, and lexical rules such as

$$\boxed{\text{dawn}}\boxed{}^+\boxed{\text{dusk}} \Rightarrow \boxed{}^+\boxed{\text{noon}}\boxed{}^+$$

(sandwiching noon between dawn and dusk) run counter to the impression given by Table 2. It is doubtful that natural language semantics has any use for irreplaceable propositions that are meant only to be observed.

In the remainder of this paper, we shift our attention from the structure of formulas (in Ψ) over to the structure of strings and languages (over the alphabet Pow(Ψ)).

5.1. Factors

Among the strings weakly subsumed by a string s are its *factors*: strings s$'$ such that s $=$ us$'$v for some (possibly null) strings u and v. It is easy to see that $L \Rightarrow L'$ is the set of strings s such that every factor of s that subsumes L also subsumes L'

$$L \Rightarrow L' = \{\mathsf{s} \in \text{Pow}(\Psi \cup \{now\})^* \mid (\text{Fac}(\mathsf{s}) \cap L^{\trianglerighteq}) \trianglerighteq L'\}$$

where Fac(s) denotes the set of factors of s

$$\text{Fac}(\mathsf{s}) = \{\mathsf{s}' \in \text{Pow}(\Psi \cup \{now\})^* \mid (\exists \mathsf{u}, \mathsf{v})\ \mathsf{u}\mathsf{s}'\mathsf{v} = \mathsf{s}\}.$$

Given a finite or infinite Ψ-schedule $x \subseteq \mathbb{Z} \times \Psi$, we define for $n, m \geq 0$, the (m, n)*th approximation of* x by restricting x to times between $-m$ and n

$$x_m^n = \{(i, \varphi) \in x \mid -m \leq i \leq n\}$$

and collect these approximations in

$$Fac(x) = \{\mathsf{s} \in now_\Psi \mid (\exists n, m \geq 0)\ sched(\mathsf{s}) = x_m^n\}\ .$$

For instance,

$$Fac(\{(i,p) \mid i \in \mathbb{Z}\}) = \Box^* \boxed{p}^* \boxed{now, p} \boxed{p}^* \Box^*.$$

For L to equal $Fac(x)$, L must also be *skeletal*: for all $\mathsf{s}, \mathsf{s}' \in L$, $\mathsf{s} \unrhd \mathsf{s}'$ implies $\mathsf{s} = \mathsf{s}'$. Formulas such as

$$\textsc{Previous}^n(\textsc{Next}^n(p)) \text{ for } n > 0$$

pose complications for building languages L equal to $Fac(x)$; to extract $\boxed{now, p}$ by applying constraints from Table 3 to

$$\Box^m \boxed{now, \textsc{Previous}^n(\textsc{Next}^n(p))} \Box^*$$

we need $m \geq n$. The obvious fix (pursued in Section4.3) is to replace $\Box^m$ by $\Box^*$. Working with a set L of *now*-anchored strings, we define $\mathrm{Fac}_{now}(L)$ to be the set of *now*-anchored strings that are factors of strings in L

$$\mathrm{Fac}_{now}(L) = \{\mathsf{s} \in now_\Psi \mid (\exists \mathsf{s}' \in L)\ \mathsf{s} \in \mathrm{Fac}(\mathsf{s}')\}$$

and collect the $\unrhd$-maximal strings of $\mathrm{Fac}_{now}(L)$ in

$$L_\infty = \unrhd\text{-max}(\mathrm{Fac}_{now}(L))$$

where

$$\unrhd\text{-max}(L') = \{\mathsf{s} \in L' \mid \text{not } (\exists \mathsf{s}' \in L' - \{\mathsf{s}\})\ \mathsf{s}' \unrhd \mathsf{s}\}.$$

It follows that

$$Sched(L) = Sched(L_\infty)$$

and if L is a path, then

$$\begin{aligned} x = Sched(L) &\iff Fac(x) = L_\infty \\ &\iff Fac(x) \equiv L_\infty \end{aligned}$$

(with $\equiv$ given by $\unrhd$ according to Section 4.1) and

$$Fac(x) = Stg(x)_\infty.$$

Moreover, a finite-state machine accepting L can be transformed into one accepting L_∞.

5.2. Parallel composition and context update

Implicit in the terminology "factors" above is a view of concatenation as multiplication. If we work with an alphabet of sets (as symbols), there is another binary operation & on languages that can equally be understood as multiplication. The *superposition* $L\&L'$ *of* languages L and L' consists of componentwise unions of strings in L and L' of the same length

$$L\&L' = \bigcup_{n \geq 0} \{(a_1 \cup b_1) \cdots (a_n \cup b_n) \mid a_1 \cdots a_n \in L \text{ and } b_1 \cdots b_n \in L'\}.$$

Whereas concatenation glues languages serially, superposition puts languages in parallel, running a finite-state machine for L in lockstep with a finite-state machine for L' to produce a finite-state machine for $L\&L'$ ([7]). Superposition lies behind the subsumption relation $\unrhd$ in Section 4 inasmuch as it reduces $\unrhd$ to containment between languages

$$L \unrhd L' \iff L \subseteq L\&L'.$$

Recalling that $L \underline{\blacktriangleright} L'$ reduces to $L \unrhd L'^{\Box}$, we can make L and L' co-occur subject to constraints C in the language

$$L\&_C L' = \left(L^{\square} \,\&\, L'^{\square}\right) \cap C.$$

C may be the constraint $\boxed{now}\,\square^*\,\boxed{now} \Rightarrow \emptyset$ intersected with any number of instances of

$$\boxed{p,\bar{p}} \Rightarrow \emptyset$$

if not perhaps

$$\square \Rightarrow \boxed{p} + \boxed{\bar{p}}.$$

Finally, we can define the *update* $C[L]$ *of* context C *by* L to be $L\&_C C$

$$\begin{aligned} C[L] &= L\&_C C \\ &= \{s \in C \mid s \underline{\blacktriangleright} L\} \end{aligned}$$

and say L *implies* L' *relative to* C, $L \vdash^C L'$, if $C[L] \underline{\blacktriangleright} L'$ ([10]).[8]

References

[1] N. Asher, A. Lascarides, Logics of Conversation, Cambridge University Press, Cambridge, MA, 2003.
[2] J. Barwise, J. Perry, Situations and Attitudes, MIT Press, Cambridge, MA, 1983.
[3] K.R. Beesley, L. Karttunen, Finite State Morphology, CSLI Publications, Stanford, CA, 2003.
[4] E.M. Clarke, O. Grumberg, D.A. Peled, Model Checking, MIT Press, Cambridge, MA, 1999.
[5] R. Cooper, Records and record types in semantic theory, Journal of Logic and Computation 15 (2) (2005) 99–112.
[6] P. Dekker, Cases, adverbs, situations and events, in: H. Kamp, B. Partee (Eds.), Context Dependence in the Analysis of Linguistic Meaning, Elsevier, Amsterdam, 2004, pp. 383–404.
[7] T. Fernando, A finite-state approach to events in natural language semantics, Journal of Logic and Computation 14 (1) (2004) 79–92.
[8] T. Fernando, Inertia in temporal modification, Semantics and Linguistic Theory XIV, Cornell Linguistics Circle Publications, 2004, pp. 56–73.
[9] T. Fernando, Comic relief for anankastic conditionals, in: Proceedings of the 15th Amsterdam Colloquium, Amsterdam, 2005, pp. 71–76.
[10] T. Fernando, Finite-state temporal projection, in: Proceedings of the 11th International Conference on Implementation and Application of Automata, Lecture Notes in Computer Science 4094, Springer, 2006, pp. 230–241.
[11] T. Fernando, Representing events and discourse: comments on Hamm, Kamp and van Lambalgen, Theoretical Linguistics 32 (1) (2006) 57–64.
[12] T. Fernando, Observing events and situations in time, Linguistics and Philosophy 30 (5) (2007) 527–550.
[13] T. Fernando, Branching from inertia worlds, Journal of Semantics 25 (3) (2008) 321–344.
[14] T. Fernando, R. Nairn, Entailments in finite-state temporality, in: Proceedings of the Sixth International Workshop on Computational Semantics, Tilburg University, 2005, pp. 128–138.
[15] H. Kamp, U. Reyle, From Discourse to Logic, Kluwer Academic Publishers, Dordrecht, 1993.
[16] A. Kratzer, An investigation of the lumps of thought, Linguistics and Philosophy 12 (1989) 607–653.
[17] M. van Lambalgen, F. Hamm, The Proper Treatment of Events, Blackwell, Oxford, 2005.
[18] J. McCarthy, P. Hayes, Some philosophical problems from the standpoint of artificial intelligence, in: M. Meltzer, D. Michie (Eds.), Machine Intelligence, vol. 4, Edinburgh University Press, Edinburgh, 1969, pp. 463–502.
[19] Y.N. Moschovakis, Sense and denotation as algorithm and value, in: J. Oikkonen, J. Vaananen (Eds.), Logic Colloquium'90, Number 2 in Lecture Notes in Logic, Springer, 1994, pp. 210–249.
[20] T. Parsons, Events in the Semantics of English: A Study in Subatomic Semantics, MIT Press, Cambridge, MA, 1990.
[21] A. Prior, Past, Present and Future, Clarendon Press, Oxford, 1967.
[22] A. Ranta, Type-Theoretical Grammar, Oxford University Press, Oxford, MA, 1994.
[23] H. Reichenbach, Elements of Symbolic Logic, Macmillan, London, 1947.
[24] L. Schubert, The situations we talk about, in: J. Minker (Ed.), Logic-Based Artificial Intelligence, Kluwer, Dordrecht, 2000, pp. 407–439.
[25] G. Sundholm, Proof theory and meaning, in: D. Gabbay, F. Guenthner (Eds.), Handbook of Philosophical Logic, vol. 3, Reidel, Dordrecht, 1986, pp. 471–506.
[26] C. Tenny, Grammaticalizing aspect and affectedness, Dissertation, Department of Linguistics and Philosophy, MIT, 1987.
[27] Z. Vendler, Linguistics in Philosophy, Cornell University Press, Ithaca, NY, 1967.
[28] S. Vickers, Topology via Logic, Cambridge University Press, Cambridge, MA, 1989.

8 For a more direct approach to the regular languages in Section 4 (proceeding essentially from Sections 5.1 and 5.2), see 'Temporal propositions as regular languages,' presented in the Sixth International Workshop on Finite-State Methods and Natural Language Processing, Potsdam, September 2007.