Finite-State Temporal Projection

Tim Fernando

Computer Science, Trinity College, Dublin 2, Ireland

Abstract. Finite-state methods are applied to determine the consequences of events, represented as strings of sets of fluents. Developed to flesh out events used in natural language semantics, the approach supports reasoning about action in AI, including the frame problem and inertia. Representational and inferential aspects of the approach are explored, centering on conciseness of language, context update and constraint application with bias.

1 Introduction

What follows, given that certain events happen? This question is addressed below through finite-state methods. Automata and their runs are basic to temporal logic (e.g. Eme92). In Linear Temporal Logic (LTL), for instance, an infinite string $x_1 x_2 \cdots$ of sets x_i of atomic propositions specifying *all* atomic propositions true at time i extends uniquely to an infinite string $\hat{x_1}\hat{x_2}\cdots$ of sets $\hat{x_i}$ of formulae true at i. To capture events that make up the string $x_1 x_2 \cdots$, it is convenient to consider fragments $a_n a_{n+1} \cdots a_{n+m}$ of $\hat{x_1}\hat{x_2}\cdots$, where $a_k \subseteq \hat{x_k}$ for $n \leq k \leq n{+}m$. These fragments cover only finite stretches of $\hat{x_1}\hat{x_2}\cdots$ and may include only some formulae true over those stretches.

In general, let Φ be a set of formulae called *fluents*. A fluent may or may not be atomic, and may or may not belong to LTL. We identify an event with a string $a_1 a_2 \cdots a_n$ over the alphabet 2^Φ of sets of fluents, drawing boxes, instead of curly braces, to enclose a set of fluents when it is intended as a symbol of the alphabet. This notation reinforces the intuition that an event is a film strip assembled from partial snapshots $\in 2^\Phi$, and helps, for instance, distinguish the string $\Box$ (that we conflate with the language $\{\Box\}$) from the empty language $\emptyset$ (containing no strings). As an event, the string $\boxed{\varphi}\Box$ is as much a part of $\boxed{\varphi}\boxed{\varphi}$ as it is of $\boxed{\varphi}\boxed{\overline{\varphi}}$, where $\overline{\varphi}$ is the negation of φ. Henceforth, we assume negation $\bar{\cdot}$ is a map on fluents such that $\overline{\overline{\varphi}} = \varphi \neq \overline{\varphi}$ for every fluent φ.

We shall see shortly that identifying an event (instance) with a string has certain defects. We will finesse the problem by focusing less on the conception of an event-as-string and more on that of an event-type-as-language. Much of what follows concerns operations and relations on languages over the alphabet 2^Φ that build up and relate event-types in useful ways.

1.1 Superposition and the Allen Interval Relations

A natural conjunction of languages L, L' over 2^Φ is the *superposition of* L *and* L' (Fer04) obtained from the componentwise union of strings in L and L' of the same length

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

For instance, we can compose a language for *rain from dawn to dusk* as

$$\boxed{\text{rain}}^+ \ \&\ \boxed{\text{dawn}}\Box^+ \ \&\ \Box^+\boxed{\text{dusk}} \ = \ \boxed{\text{rain, dawn}}\boxed{\text{rain}}^*\boxed{\text{rain, dusk}}$$

(suppressing parentheses, as & is associative) consisting for $n \geq 0$ of strings $\boxed{\text{rain, dawn}}\boxed{\text{rain}}^n\boxed{\text{rain, dusk}}$ with $(n+2)$ snapshots, all of rain, the first at dawn, and the last at dusk. The different values of n support models at different levels of temporal granularity (the larger the n, the finer the grain). It is not obvious, however, that we should think of each of these infinitely many strings as distinct events. And surely we can reduce the infinite language to some finite core that captures its essence: three snapshots, $\boxed{\text{rain, dawn}}$, $\boxed{\text{rain}}$ and $\boxed{\text{rain, dusk}}$, arranged in a particular order.

Indeed, it is tempting to reduce every string $\boxed{\text{rain, dawn}}\boxed{\text{rain}}^n\boxed{\text{rain, dusk}}$ with $n \geq 1$ to the string $\boxed{\text{rain, dawn}}\boxed{\text{rain}}\boxed{\text{rain, dusk}}$ of length 3. More generally, we define the *interval reduction* $\mathsf{ir}(s)$ of a string $s \in (2^{\Phi})^*$ inductively

$$\mathsf{ir}(s) \overset{\text{def}}{=} \begin{cases} s & \text{if length}(s) \leq 1 \\ \mathsf{ir}(as') & \text{if } s = aas' \\ a\,\mathsf{ir}(a's') & \text{if } s = aa's' \text{ where } a \neq a' \end{cases}$$

(for all $a, a' \subseteq \Phi$), reducing a block aa of two a's to one, in line with the dictum "no time without change" (KR93, page 674). What does the modification "interval" in ir have to do with Allen's 13 interval relations, tabulated below?

p before q	$\boxed{p}\Box\boxed{q}$	p after q	$\boxed{q}\Box\boxed{p}$
p meets q	$\boxed{p}\boxed{q}$	p met-by q	$\boxed{q}\boxed{p}$
p overlaps q	$\boxed{p}\boxed{p,q}\boxed{q}$	p overlapped-by q	$\boxed{q}\boxed{p,q}\boxed{p}$
p starts q	$\boxed{p,q}\boxed{q}$	p started-by q	$\boxed{p,q}\boxed{p}$
p during q	$\boxed{q}\boxed{p,q}\boxed{q}$	p contains q	$\boxed{p}\boxed{p,q}\boxed{p}$
p finishes q	$\boxed{q}\boxed{p,q}$	p finish-by q	$\boxed{p}\boxed{p,q}$
p equals q	$\boxed{p,q}$		

It turns out we can extract each of the 13 strings in the language

$$\begin{aligned}\text{Allen}(p,q) \overset{\text{def}}{=}\ & \boxed{p}(\epsilon+\Box)\boxed{q} \ + \ \boxed{q}(\epsilon+\Box)\boxed{p} \ + \\ & (\boxed{p}+\boxed{q}+\epsilon)\boxed{p,q}(\boxed{p}+\boxed{q}+\epsilon)\end{aligned}$$

(where ϵ is the null string) from the superposition

$$\Box^+\boxed{p}^+\Box^+ \ \&\ \Box^+\boxed{q}^+\Box^+ \ = \ \mathsf{ir}^{-1}(\Box\boxed{p}\Box)\ \&\ \mathsf{ir}^{-1}(\Box\boxed{q}\Box))$$

by applying ir

$$\mathsf{ir}(\mathsf{ir}^{-1}(\square\boxed{p}\square)\ \&\ \mathsf{ir}^{-1}(\square\boxed{q}\square)) \quad = \quad \square\text{Allen}(p,q)\square$$

where $\mathsf{ir}(L) \overset{\text{def}}{=} \{\mathsf{ir}(s) \mid s \in L\}$. Without the intervention of ir^{-1},

$$\mathsf{ir}(\square\boxed{p}\square\ \ \&\ \ \square\boxed{q}\square) \quad = \quad \mathsf{ir}(\square\boxed{p,q}\square) \quad = \quad \square\boxed{p,q}\square$$

and a lot of structure gets lost.[1] Going back to $\boxed{\text{rain, dawn}}\boxed{\text{rain}}^*\boxed{\text{rain, dusk}}$, we may, anticipating further superpositions, wish to live with its infinitely many strings, whether or not they correspond to distinct events.

1.2 Subsumption and Constraints

If we &-superpose $\boxed{\text{rain, dawn}}\boxed{\text{rain}}^*\boxed{\text{rain, dusk}}$ with $\square^+\boxed{\text{noon}}\square^+$, we get

$$\boxed{\text{rain, dawn}}\boxed{\text{rain}}^*\boxed{\text{rain,noon}}\boxed{\text{rain}}^*\boxed{\text{rain, dusk}}$$

which filters out the string $\boxed{\text{rain, dawn}}\boxed{\text{rain, dusk}}$ of length 2, and fleshes out the remaining strings in $\boxed{\text{rain, dawn}}\boxed{\text{rain}}^+\boxed{\text{rain, dusk}}$ by including noon in the middle. To capture the growth of information here, let us say that L *subsumes* L' and write $L \trianglerighteq L'$ if the superposition of L and L' includes L

$$L \trianglerighteq L' \quad \overset{\text{def}}{\Longleftrightarrow} \quad L \subseteq L\&L'$$

(roughly: L is at least as informative as L'). Conflating a string s with the singleton language $\{s\}$, it follows that L subsumes L' exactly if each string in L subsumes some string in L'

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

where $\trianglerighteq$ holds between strings of the same length related componentwise by inclusion

$$a_1 a_2 \cdots a_n \trianglerighteq b_1 b_2 \cdots b_m \quad \text{iff} \quad n = m \text{ and } a_i \supseteq b_i \text{ for } 1 \leq i \leq n\ .$$

For example, $\boxed{p,q} \ \trianglerighteq\ \boxed{p} \ \trianglerighteq\ \boxed{p} + \boxed{q}$. As a type with instances $s \in L$, a language L is essentially a disjunction $\bigvee_{s\in L} s$ of conjunctions s (as is clear from the model-theoretic interpretations spelled out in Fer04).

[1] Another way of making sense of Allen's relations in the present set-up is to replace intervals i and j by languages L and L' (respectively), and then turn *before*(i,j) to $L\square^+L'$, *meets*(i,j) to LL', *equals*(i,j) to $L\&L'$, *finishes*(i,j) to $\square^+L\ \&L'$, etc. These constructions are all finite-state.

Pausing to consider the finite-state character of the preceding notions, note that for finite Φ and $L \subseteq (2^{\Phi})^*$, we can construct a finite-state transducer for the relation

$$\&_L \overset{\text{def}}{=} \{(s_1, s_2) \mid s_2 \in s_1 \& L\}$$

provided L is regular. (Given a finite automaton for L, form a transducer with the labeled state transitions

$$q \xrightarrow{a:b} q' \overset{\text{def}}{\iff} (\exists c)\ q \xrightarrow{c} q' \text{ and } a \cup c = b$$

with initial and final states unchanged.) Hence, if L and L' are regular, then so is

$$L \& L' = \{s_2 \mid (\exists s_1 \in L')\ (s_1, s_2) \in \&_L\} .$$

The relation

$$\{(s, s') \mid s' \trianglerighteq s\} = \&_{(2^{\Phi})^*}$$

is also regular. Moreover, if L is regular, so is the *subsumption closure* $L^{\trianglerighteq}$ *of* L

$$L^{\trianglerighteq} \overset{\text{def}}{=} \{s \mid s \trianglerighteq L\} = L \& (2^{\Phi})^*$$

consisting of strings that subsume some string in L.

In addition to the unary operation mapping a language L to $L^{\trianglerighteq}$, subsumption $\trianglerighteq$ induces a binary operation $\Rightarrow$ on languages. Given L and L', let the *constraint* $L \Rightarrow L'$ be the set of strings s such that whenever $s \trianglerighteq \Box^n L \Box^m$, $s \trianglerighteq \Box^n L' \Box^m$

$$L \Rightarrow L' \overset{\text{def}}{=} \{s \in (2^{\Phi})^* \mid (\forall n, m \geq 0)\ s \trianglerighteq \Box^n L \Box^m \text{ implies } s \trianglerighteq \Box^n L' \Box^m\} .$$

As explained in FN05, $\Rightarrow$ is adapted from a similar construct called *restriction* in BK03, with

$$L \Rightarrow L' = \overline{(2^{\Phi})^* \ (L^{\trianglerighteq} \cap \overline{L'^{\trianglerighteq}})\ (2^{\Phi})^*}$$

where $\overline{L}$ is the set-theoretic complement $(2^{\Phi})^* - L$. (To make sense of the expression to the right, recall the Boolean equivalence between $A \supset B$ and $\neg(A \wedge \neg B)$; the counterexamples in $L \Rightarrow L'$ corresponding to $A \wedge \neg B$ are strings with substrings that subsume L but not L' — that is, substrings from $L^{\trianglerighteq} \cap \overline{L'^{\trianglerighteq}}$.) We can use $\Rightarrow$ to define the φ*-bivalent* language

$$(\varphi\text{-biv}) \qquad \Box \ \Rightarrow \ \boxed{\varphi} + \boxed{\overline{\varphi}}$$

consisting of strings $a_1 a_2 \cdots a_n \in (2^{\Phi})^*$ such that for each i from 1 to n (inclusive), $\varphi \in a_i$ or $\overline{\varphi} \in a_i$. While we may want to work with strings that do not belong to this language (allowing a string to be silent on φ), it makes sense to restrict our events to strings in the φ*-consistent* language

(φ-con) $$\boxed{\varphi, \overline{\varphi}} \Rightarrow \emptyset$$

requiring that no symbol in a string contain both φ and its negation $\overline{\varphi}$. Similarly, to pick out a unique position in a string by a fluent r (e.g. r = speech time "now" or some other reference point), the constraint

$$\boxed{\mathrm{r}}\square^{*}\boxed{\mathrm{r}} \Rightarrow \emptyset$$

precludes the occurrence of r in two different positions in a string. Again, observe that in general $L \Rightarrow L'$ is regular if L and L' are (and Φ is finite).

1.3 Inertia, Force and STRIPS Actions

Next, given a fluent φ, we introduce a fluent $\mathsf{f}\varphi$ to mark the application of a force to make φ true (at the next step). Hence, the fluent $\mathsf{f}\overline{\varphi}$ says a force is applied to make φ false. Accordingly, the constraint

$$\boxed{\varphi}\square \Rightarrow \square\boxed{\varphi} + \boxed{\mathsf{f}\overline{\varphi}}\square \tag{1}$$

states that φ persists (forwards) unless some force is applied against it, while

$$\square\boxed{\varphi} \Rightarrow \boxed{\varphi}\square + \boxed{\mathsf{f}\varphi}\square \tag{2}$$

states that φ persists backward unless it was previously forced. (1) and (2) are similar to inertial constraints formulated in FN05 except that we distinguish $\mathsf{f}\varphi$ from $\mathsf{f}\overline{\varphi}$ here (the previous fluents $\mathsf{F}\varphi$ amounting essentially to $\mathsf{f}\varphi \vee \mathsf{f}\overline{\varphi}$) in order to formulate the constraint

$$\boxed{\mathsf{f}\varphi}\square \Rightarrow \square\boxed{\varphi} + \boxed{\mathsf{f}\overline{\varphi}}\square \tag{3}$$

saying an unopposed force on φ brings φ about at the next moment. We could derive (1) from (3) and the constraint

$$\boxed{\varphi} \Rightarrow \boxed{\mathsf{f}\varphi} \tag{4}$$

but (4) would, under ($\mathsf{f}\varphi$-con), rule out snapshots $\boxed{\varphi, \overline{\mathsf{f}\varphi}, \mathsf{f}\overline{\varphi}}$ that make $\overline{\varphi}$ true at the next step, assuming (3) for $\overline{\varphi}$. Instead of (4), we use $\mathsf{f}\varphi$ to encode the representation of an action A in STRIPS (FN71) by the constraints

$$\boxed{\text{try(A)}} \Rightarrow \boxed{\mathsf{f}\varphi_1, \ldots \mathsf{f}\varphi_n} \text{ where Add-List(A)} = [\varphi_1, \ldots, \varphi_n]$$

$$\boxed{\text{try(A)}} \Rightarrow \boxed{\mathsf{f}\overline{\psi_1}, \ldots, \mathsf{f}\overline{\psi_m}} \text{where Delete-List(A)} = [\psi_1, \ldots, \psi_m]$$

alongside

$$\boxed{\text{try(A)}} \Rightarrow \boxed{\chi_1, \ldots, \chi_k} \text{ where Precondition-List(A)} = [\chi_1, \ldots, \chi_k]$$

(borrowing the notation try(A) from AF94, with the temporal parameter implicit). In STRIPS, only one action is performed at a time, with deterministic post-conditions provided by constraints of the form (3) asserting that an unopposed force succeeds. The constraints above go beyond STRIPS in allowing multiple actions to execute simultaneously, and the effects of an action to be non-deterministic.

1.4 The Remainder of This Paper

Building on §§1.1 and 1.2 above (which draw on Fer04 and FN05, in pursuit of a line of research derived from Ste00), we consider the conciseness of representation in §2, the contribution context as a language makes to entailments in §3, and the application of constraints for inference rules in §4.

2 Concise Representations and Minimal Strings

In this section, we examine the conciseness of our string/language representations, starting with an observation worked out in Kar05 that we can unpack the symbols in our alphabet 2^{Φ} as strings. In a string $a_1 \cdots a_n \in (2^{\Phi})^*$, each addition a_{i+1} to $a_1 \cdots a_i$ is understood as describing a succeeding moment of time. But, of course, the sequential structure in a string need not mark the passage of time. Instead, assuming Φ is finite, as henceforth we do, we can define a surjective function $\pi : \Phi^* \to 2^{\Phi}$ by

$$\begin{aligned} \pi(\epsilon) &\overset{\text{def}}{=} \square \\ \pi(\varphi s) &\overset{\text{def}}{=} \boxed{\varphi} \cup \pi(s) \end{aligned}$$

so for example $\boxed{\varphi, \psi} = \pi(\varphi\psi) = \pi(\varphi\psi\varphi)$. We then introduce a new symbol "tick" $\wr \notin \Phi^+$ to advance the clock so that we can encode, for instance, $\boxed{\varphi, \psi}\boxed{\varphi}$ as the string $\varphi\psi \wr \varphi$, or as $\psi\varphi\psi \wr \varphi\varphi$.

2.1 Snapshots-as-Symbols Versus Snapshots-as-Strings

Given an ordering of the fluents, we can pick out for each $a \in 2^{\Phi}$ a canonical representative $\hat{\pi}(a) \in \Phi^*$ such that $\pi(\hat{\pi}(a)) = a$. Extending $\hat{\pi}$ to strings over 2^{Φ}, we can define $\hat{\pi} : (2^{\Phi})^* \to (\Phi \cup \{\wr\})^*$ by

$$\begin{aligned} \hat{\pi}(\epsilon) &\overset{\text{def}}{=} \epsilon \\ \hat{\pi}(as) &\overset{\text{def}}{=} \hat{\pi}(a) \wr \hat{\pi}(s) \end{aligned}$$

so for example $\hat{\pi}(a_1 a_2 a_3) = \hat{\pi}(a_1) \wr \hat{\pi}(a_2) \wr \hat{\pi}(a_3) \wr$. It is easy to build a finite-state transducer for $\{(s, \hat{\pi}(s)) \mid s \in (2^{\Phi})^*\}$.

Proposition 1. *If $L \subseteq (2^{\Phi})^*$ is regular then so is*

$$L_{\hat{\pi}} \overset{\text{def}}{=} \{\hat{\pi}(s) \mid s \in L\} .$$

Indeed, if $R \subseteq (2^{\Phi})^* \times (2^{\Phi})^*$ is regular, so is

$$R_{\hat{\pi}} \overset{\text{def}}{=} \{(\hat{\pi}(s), \hat{\pi}(s')) \mid sRs'\} .$$

Proof. $R_{\hat{\pi}} = \hat{\pi}^{-1}; R; \hat{\pi}$ where ; is sequential composition. $\dashv$

By Proposition 1, we can work with the alphabet 2^{Φ} without worrying if regularity is preserved when switching over to the alphabet $\Phi \cup \{\wr\}$. Equally, a regular relation R over the alphabet $\Phi \cup \{\wr\}$ remains regular when translated to $\hat{\pi}; \mathsf{R}; \hat{\pi}^{-1} \subseteq (2^{\Phi})^* \times (2^{\Phi})^*$. The choice between snapshots-as-symbols and snapshots-as-strings is a matter of taste, as far as regularity is concerned. From a processing perspective, it is noteworthy that working with $\Phi \cup \{\wr\}$ makes greater use of the finite-state machinery. But for that very reason, the theory is arguably simpler to describe in terms of 2^{Φ} (with the unwinding of a snapshot to a string over $\Phi \cup \{\wr\}$ kept in the background).

2.2 Minimal Strings and Weak Subsumption

Given a language L and string s over 2^{Φ}, let us call s $\trianglerighteq$-*minimal in* L if $s \in L$ and for all $s' \in L - \{s\}$, not $s \trianglerighteq s'$. Let $L_{\trianglerighteq}$ be the set of strings $\trianglerighteq$-minimal in L

$$L_{\trianglerighteq} \overset{\text{def}}{=} \{s \in L \mid (\forall s' \in L - \{s\}) \text{ not } s \trianglerighteq s'\} \ .$$

For example, $(2^{\Phi})_{\trianglerighteq} = \Box$, and recalling the constraint (φ-biv) of φ-bivalence

$$(\Box \Rightarrow \boxed{\varphi} + \boxed{\overline{\varphi}})_{\trianglerighteq} \ = \ (\boxed{\varphi} + \boxed{\overline{\varphi}})^*$$

or in words: $(\boxed{\varphi} + \boxed{\overline{\varphi}})^*$ is the set of $\trianglerighteq$-minimal φ-bivalent strings.

Proposition 2. *Let L be a language over the alphabet 2^{Φ}.*

(a) $L_{\trianglerighteq}$ is the $\subseteq$-least language $\trianglerighteq$-equivalent[2] *to L, while $L^{\trianglerighteq}$ is the $\subseteq$-greatest.*
(b) If L is regular, then so are $L_{\trianglerighteq}$ and $L^{\trianglerighteq}$.

Proof. Straightforward, where a finite automaton for L can be turned into one for $\{s \mid (\exists s' \in L)\ s \trianglerighteq s' \text{ and } s \neq s'\}$, making

$$L_{\trianglerighteq} \ = \ L - \{s \mid (\exists s' \in L)\ s \trianglerighteq s' \text{ and } s \neq s'\}$$

regular (by the closure properties of regular languages). $\dashv$

We can minimize L further by "unpadding" $L_{\trianglerighteq}$. More precisely, for every string $s \in (2^{\Phi})^*$, let $\mathsf{unpad}(s)$ be s with all initial and final $\Box$'s stripped off

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

For example, $\mathsf{unpad}(\Box\Box\boxed{\varphi}\Box\boxed{\psi}\Box) = \boxed{\varphi}\Box\boxed{\psi}$. Next, for every language L over 2^{Φ}, let $\mathsf{unpad}(L) \overset{\text{def}}{=} \{\mathsf{unpad}(s) \mid s \in L\}$ and call L *unpadded* if $\mathsf{unpad}(L) = L$. Let $L^{\Box}$ consist of all strings in L with any number of leading and trailing $\Box$'s deleted or added

$$L^{\Box} \overset{\text{def}}{=} \Box^* \mathsf{unpad}(L) \Box^* \ = \ \{s \mid \mathsf{unpad}(s) \in \mathsf{unpad}(L)\} \ .$$

[2] Given a relation $\mathcal{R}$ (such as $\trianglerighteq$) between languages, we say L is $\mathcal{R}$-*equivalent* to L' if $L\mathcal{R}L'$ and $L'\mathcal{R}L$.

Incorporating unpadding into subsumption $\trianglerighteq$, let *weak subsumption* $\underline{\blacktriangleright}$ be $\trianglerighteq$ with the second argument L' weakened to $L'^{\Box}$

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

Weak subsumption $\underline{\blacktriangleright}$ compares information content in the same way as $\trianglerighteq$

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

but without insisting that strings have the same length

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

Insofar as $L \underline{\blacktriangleright} L'$ says every instance of L contains some instance of L', it is natural to say L' *happens in* L when $L \underline{\blacktriangleright} L'$. For the record, we have

Proposition 3. *Let $L \subseteq (2^{\Phi})^*$.*

(a) $\mathsf{unpad}(L_{\trianglerighteq})$ *is the $\subseteq$-least unpadded language $\underline{\blacktriangleright}$-equivalent to L.*
(b) The relations $\{(s, \mathsf{unpad}(s)) \mid s \in (2^{\Phi})^*\}$ and $\{(s, s') \mid s' \underline{\blacktriangleright} s\}$ are regular. Hence, $\mathsf{unpad}(L)$ *and $L^{\Box}$ are regular if L is.*

3 Context as Language

It is a truism that information should be understood relative to context. But what does this mean for information and context represented as languages over the alphabet 2^{Φ}? If we regard strings over 2^{Φ} as epistemic possibilities much like possible worlds except that strings are finite and their snapshots non-exhaustive, then we can turn weak subsumption $\underline{\blacktriangleright}$ into an inference relation $\vdash^{C}$ factoring in background information encoded by a context $C \subseteq (2^{\Phi})^*$. Let

$$L \vdash^{C} L' \overset{\text{def}}{\iff} C[L] \underline{\blacktriangleright} L'$$

for some notion $C[L] \subseteq (2^{\Phi})^*$ of C *updated by* L. Precisely what $C[L]$ might be depends on what we assume about C and L. In this section, we will assume C describes the "big picture" with *global* constraints such as $\boxed{\varphi, \overline{\varphi}} \Rightarrow \emptyset$ providing the background for the more *local* events (with bounded temporal extents) described by L. For example, we might expect an event $L = \boxed{\text{rain,now}}$ of *raining now* to update a background

$$C \;=\; (\Box + \boxed{\text{rain}} + \boxed{\overline{\text{rain}}})^{+} \ \& \ (\Box^{+}\boxed{\text{now}}\Box^{+})$$

to give

$$C[L] \;=\; (\Box + \boxed{\text{rain}} + \boxed{\overline{\text{rain}}})^{*}\boxed{\text{rain,now}}(\Box + \boxed{\text{rain}} + \boxed{\overline{\text{rain}}})^{*} \ .$$

The asymmetry assumed above between C and L means that neither the intersection $C \cap L$ nor the superposition $C \& L$ will do for $C[L]$.

3.1 Context Updated by L

Keeping the previous example in mind, let us define

$$C[L] \overset{\text{def}}{=} \{s \in C \mid s \underline{\blacktriangleright} L\}$$

thereby ensuring that the update $C[L]$ preserves C and L. We write $s \in C$ for "s complies with the global constraint C," and $s \underline{\blacktriangleright} L$ for "s complies with the local conditions L" (admittedly, a rather long-winded way of saying L happens in s).

Proposition 4. *Let $C, L \subseteq (2^{\Phi})^*$.*

(a) $C[L] \underline{\blacktriangleright} L$ and $C[L] = (C[L])[L]$ and $C[C] = C$.
(b) $C[L] = \bigcup_{s \in L} C[s]$ and the relations

$$\begin{aligned} \{(s,s') \mid s' \underline{\blacktriangleright} s \text{ and } s' \in C\} &= \{(s,s') \mid s' \underline{\blacktriangleright} s\} \ ; \ \{(s',s') \mid s' \in C\} \\ \{(s,s') \in L \times C \mid s' \underline{\blacktriangleright} s\} &= \{(s,s) \mid s \in L\} \ ; \ \{(s,s') \mid s' \underline{\blacktriangleright} s \text{ and } s' \in C\} \end{aligned}$$

and language $C[L]$ are regular if C and L are.
(c) $(C[L])[L'] = (C[L'])[L]$ for any $L' \subseteq (2^{\Phi})^*$.
(d) $C[L] = C[L']$ for every language L' $\underline{\blacktriangleright}$-equivalent to L. In particular, $C[L] = C[L_{\trianglerighteq}]$.
(e) $C[L] = (L^{\Box} \& C) \cap C$. Hence, if $C = C^{\trianglerighteq}$, then $C[L] = L^{\Box} \& C$.

Proof. All parts are routine, with the regularity in part (b) a corollary of Proposition 3(b), part (d) following from the biconditional

$$s \underline{\blacktriangleright} L \quad \text{iff} \quad s \underline{\blacktriangleright} L'$$

for every L' $\underline{\blacktriangleright}$-equivalent to L, and part (e) from

$$C \underline{\blacktriangleright} L \quad \text{iff} \quad C \subseteq L^{\Box} \& C \ .$$

$\dashv$

3.2 The C-Negation of L

What part of a context C would be *un*able to support an update by L? Calling that part $\text{neg}_C L$, we require that $(\text{neg}_C L)[L] = \emptyset$. An obvious candidate for $\text{neg}_C L$ is $C - C[L]$. The problem is that for say, $C = (\Box + \boxed{p} + \boxed{\overline{p}})^+$,

$$\begin{aligned} C[\boxed{p}] &= (\Box + \boxed{p} + \boxed{\overline{p}})^* \boxed{p} (\Box + \boxed{p} + \boxed{\overline{p}})^* \\ C - C[\boxed{p}] &= (\Box + \boxed{\overline{p}})^+ \\ (C - C[\boxed{p}])[\boxed{p}] &= (\Box + \boxed{\overline{p}})^* \boxed{p} (\Box + \boxed{\overline{p}})^* \quad \neq \ \emptyset \ . \end{aligned}$$

So instead, let us define $\text{neg}_C L$ to be the $\subseteq$-largest sublanguage L' of C such that $L' \& L^{\Box}$ is disjoint from C

$$\text{neg}_C L \overset{\text{def}}{=} \{s \in C \mid (s \& L^{\Box}) \cap C = \emptyset\} \ .$$

Proposition 5. *Let $C, L \subseteq (2^{\Phi})^*$.*

(a) For any $L' \subseteq (2^{\Phi})^*$, the following are equivalent.
  (i) $C[L'] \subseteq neg_C L$
  (ii) $(C[L'])[L] = \emptyset$
  (iii) $(C[L])[L'] = \emptyset$
  (iv) $C[L] \subseteq neg_C L'$
(b) $neg_C L$ *is regular if* C *and* L *are.*

Proof. For part (a), the equivalence of (ii) and (iii) follows from the commutativity expressed in Proposition 4(c). Note that (i) fails iff for some $s \in C[L']$, there is an $s' \in (s\&L^{\Box}) \cap C$. Such a pair s, s' is what is precisely needed for (ii) to fail, by Proposition 4(b,e).

For part (b), the trick, as with $L_{\trianglerighteq}$, is to use the closure of regular languages under complementation

$$\text{neg}_C L \ = \ C - \{s \in C \mid (\exists s' \in L^{\Box})(\exists s'' \in C) \ s\&s' \simeq s''\}$$

where $s_1\&s_2 \simeq s_3$ abbreviates $\{s_1\}\&\{s_2\} = \{s_3\}$. $\dashv$

Armed with the definition of $\text{neg}_C(L)$ above, we can for many languages L associate a complement $\neg L$ such that for contexts C meeting minimal conditions, $C[\neg L] = \text{neg}_C L$. More in §4.3 below.

4 Constraints as Inference Rules

If the previous section is about changes to context when a language is asserted or denied, this (final) section is about changes to a language L when constraints C from context are applied to L. We start by injecting L into C

$$\text{inject}(L, C) \ \stackrel{\text{def}}{=} \ \{s \in C \mid s \trianglerighteq L\} \ = \ L^{\trianglerighteq} \cap C$$

(so $C[L] = \text{inject}(L^{\Box}, C)$) before taking the strings $\trianglerighteq$-minimal in that set for the result of applying C to L

$$\text{apply}(C, L) \ \stackrel{\text{def}}{=} \ (\text{inject}(L, C))_{\trianglerighteq} \ .$$

Recall from §3.1 that L is presumed to be local, and from Proposition 4(d) that we can reduce L to $L_{\trianglerighteq}$. If C has the form $L \Rightarrow L'$, then $\text{apply}(L \Rightarrow L', L)$ looks very much like the rule of inference *modus ponens* and indeed we will see that $\text{apply}(L \Rightarrow L', L) \trianglerighteq L'$. We will proceed more generally, allowing the lefthand side of C to differ (or not differ) from L, and paying special attention to the case where the righthand side of C has the form $B_1 + B_2$.

4.1 Minimal Changes

For a constraint C of the form $A \Rightarrow B$, let us define a relation $\langle\!\langle A, B \rangle\!\rangle$ between strings where fragments in A are superposed with strings in B

$$s\langle\!\langle A, B \rangle\!\rangle s' \ \stackrel{\text{def}}{\iff} \ (\exists x \in A, y \in B, n, m \geq 0) \ s \trianglerighteq \Box^n x \Box^m, \ s\&\Box^n y \Box^m \simeq s'.$$

Let $\langle\!\langle A, B \rangle\!\rangle^*$ be the reflexive transitive closure of $\langle\!\langle A, B \rangle\!\rangle$.

Proposition 6. *For $A, B, L \subseteq (2^{\Phi})^*$, apply$(A \Rightarrow B, L)$ is the set of strings $\trianglerighteq$-minimal in*

$$\{s' \in A \Rightarrow B \mid (\exists s \in L)\ s\ \langle\!\langle A, B \rangle\!\rangle^*\ s'\}\ .$$

It is regular if A, B and L are.

4.2 Biased Changes

Suppose further that the constraint C has the form $A \Rightarrow (B_1 + B_2)$, such as the constraints (φ-biv) and (1)-(3) from §1.3. We can arrange a language L to satisfy C by adding, as stated in Proposition 6, fragments of B_1 or of B_2. Introducing a bias for B_1, let us restrict $\langle\!\langle A, B_1 + B_2 \rangle\!\rangle$ to a variant $\langle\!\langle A, B_1, B_2 \rangle\!\rangle$ that can be read: "if A then add B_1 unless B_2"

$$\begin{aligned} s\ \langle\!\langle A, B_1, B_2 \rangle\!\rangle\ s' \ \overset{\text{def}}{\iff}\ & (\exists x \in A, y \in B_1, n, m \geq 0)\ s \trianglerighteq \Box^n x \Box^m, \\ & s \ntrianglerighteq \Box^n B_2 \Box^m \text{ and } s\&\ \Box^n y \Box^m \simeq s'\ . \end{aligned}$$

To implement this restriction, we must constrain the step to $L^{\trianglerighteq}$ in apply(C, L) to satisfy C, introducing a new parameter L' to define

$$\begin{aligned} \text{app}(C, L, L') &\overset{\text{def}}{=} ((L\&L') \cap C)_{\trianglerighteq} \\ \text{ap}(A, B_1, B_2, L) &\overset{\text{def}}{=} \text{app}(A \Rightarrow (B_1 + B_2), L, (2^{\Phi(B_1)})^*) \end{aligned}$$

where $\Phi(B_1)$ is the set $\{\varphi \in \Phi \mid (\exists s \in B_1)\ s \trianglerighteq \Box^* \boxed{\varphi} \Box^*\}$ of fluents occurring in B_1. Notice that for the constraints (φ-biv) and (1)-(3), $\Phi(B_1)$ adds nothing to B_2 inasmuch as B_2 is disjoint from $\overline{B_2}\ \&\ (2^{\Phi(B_1)})^*$.

Proposition 7. *For $A, B_1, B_2, L \subseteq (2^{\Phi})^*$, ap$(A, B_1, B_2, L)$ is the set of strings $\trianglerighteq$-minimal in*

$$\{s' \in A \Rightarrow (B_1 + B_2) \mid (\exists s \in L)\ s\ \langle\!\langle A, B_1, B_2 \rangle\!\rangle^*\ s'\}$$

assuming B_2 is disjoint from $\overline{B_2}\&(2^{\Phi(B_1)})^*$. The language ap$(A, B_1, B_2, L)$ is regular if A, B_1, B_2 and L are.

4.3 Refinements Based on Consistency

Bias applied blindly can lead to trouble, as illustrated by the case of (1)

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

in the presence of the consistency constraint

$$\Phi\text{-con} \overset{\text{def}}{=} \bigcap_{\psi \in \Phi} (\boxed{\psi, \overline{\psi}} \Rightarrow \emptyset)\ \ =\ \ \{a \subseteq \Phi \mid (\forall \psi \in a)\ \overline{\psi} \notin a\}^*\ .$$

When imposing (1) on $\boxed{\varphi}\boxed{\overline{\varphi}}$, we must take the second disjunct $\boxed{\mathsf{f}\overline{\varphi}}\Box$ on the righthand side of (1). But on $\boxed{\varphi, \overline{\mathsf{f}\overline{\varphi}}}\Box$, we must opt for the first disjunct $\Box\boxed{\varphi}$.

To cover such contingencies, let us refine our transformation ap(A, B_1, B_2, L) of L, rewriting $A \Rightarrow (B_1 + B_2)$ as two constraints,

$$(A\&\neg B_1) \Rightarrow B_2 \quad \text{and} \quad (A\&\neg B_2) \Rightarrow B_1$$

where $\neg B_1$ and $\neg B_2$ are defined from Φ-con and A as follows. Let A-con $\overset{\text{def}}{=}$ $A^{\trianglerighteq} \cap \Phi$-con and

$$\neg B_i \overset{\text{def}}{=} \{s \in A\text{-con} \mid (s\&B_i) \cap A\text{-con} = \emptyset\}_{\trianglerighteq}$$

for $i \in \{1, 2\}$. (So for (1), $\neg B_1 = \boxed{\varphi}\boxed{\overline{\varphi}}$ and $\neg B_2 = \boxed{\varphi, \overline{\mathsf{f}\overline{\varphi}}}\Box$.) Continuing to keep the notation simple, we form in sequence the languages

$$\begin{aligned} L_1 &\overset{\text{def}}{=} \text{apply}((A\&\neg B_1) \Rightarrow B_2, L) \\ L_2 &\overset{\text{def}}{=} \text{ap}(A, \neg B_2, B_2, L_1) \\ L_3 &\overset{\text{def}}{=} \text{apply}((A\&\neg B_2) \Rightarrow B_1, L_2) \end{aligned}$$

which are regular if A, B_1, B_2 and L are. The idea is to settle the question of B_2 *versus* $\neg B_2$ given A, minimizing B_2 in L_2 before reducing $A \Rightarrow (B_1 + B_2)$ to $(A\&\neg B_2) \Rightarrow B_1$ in L_3. The bias favoring B_1 in $A \Rightarrow (B_1 + B_2)$ is derived from a bias favoring $\neg B_2$ in $A \Rightarrow (\neg B_2 + B_2)$ after giving B_2 its due in L_1. The general point is to subject the non-determinism in constraints $A \Rightarrow (B + B')$ to some default preference between B and $\neg B$.

References

Allen, J.F., Ferguson, G.: Actions and events in interval temporal logic. *Journal of Logic and Computation*, 4(5):531–579, 1994.

Beesley, K.R., Karttunen, L.: *Finite State Morphology.* CSLI, Stanford, 2003.

Emerson, E.A.: Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Methods and Semantics, pages 995–1072. MIT Press, 1992.

Fernando, T.: A finite-state approach to events in natural language semantics. *Journal of Logic and Computation*, 14(1):79–92, 2004.

Fernando, T., Nairn, R.: Entailments in finite-state temporality. In *Proc. 6th International Workshop on Computational Semantics*, pages 128–138. Tilburg University, 2005.

Fikes, R.E., Nilsson, N.J.: STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2:189–208, 1971.

Kamp, H., Reyle, U.: *From Discourse to Logic.* Kluwer, Dordrecht, 1993.

Karttunen, L.: `www.stanford.edu/~laurik/fsmbook/examples/YaleShooting.html`, 2005.

Steedman, M.: *The Productions of Time.* Draft, `ftp://ftp.cogsci.ed.ac.uk/pub/steedman/temporality/temporality.ps.gz`, July 2000. Subsumes 'Temporality,' in J. van Benthem and A. ter Meulen, editors, *Handbook of Logic and Language*, pages 895–935, Elsevier North Holland, 1997.