Action Signatures and Finite-State Variations

Tim Fernando
School of Computer Science and Statistics
Trinity College Dublin, Ireland
Tim.Fernando@tcd.ie

Abstract

Finite fragments of action signatures from Gelfond and Lifschitz (1998) are modified for use as bounded granularities for natural language semantics, subject to variation. Strings formed from these signatures are expanded to finite-state representations, which are compared with action languages in the tradition of Gelfond and Lifschitz.

1 Introduction

To characterise actions as labels on transitions between states which assign values to fluents, a number of definitions are collected in Gelfond and Lifschitz (1998). These apply to a range of action languages shaped by logic programming, the idea being that "a high-level action language is used as a front end for a logic programming system description" (Inclezan and Gelfond, 2016, page 189). That said, an instance of an action language predating logic programming can be found in Kleene (1956), where finite automata were introduced to represent events in neural nets from McCulloch and Pitts (1943). Those automata were subsequently simplified in Rabin and Scott (1959), a paper on which textbook presentations of finite automata are based (and singled out in its authors' 1976 Turing Award citation). As fruitful as that simplification has been, it leaves out two ingredients from Kleene (1956) that link the automata there not only to McCulloch-Pitts neurons but also to action languages, viz. inner cells and input cells. From these cells spring fluents, values and (elementary) actions that make up an *action signature* in Gelfond and Lifschitz (1998). Given an action signature, Gelfond and Lifschitz define transition systems from which chains of transitions are formed, called histories.

Skipping over transition systems, the present paper approximates histories with strings, adopting the perspective of *grammatical inference* (e.g., Heinz and Sempere, 2016; de la Higuera, 2010) to view strings as (known) data points to be explained by (unknown) automata (essential components of which include transitions). Applied to linguistic semantics, the intuition is that an episodic report such as

(1) Facebook bought Instagram

is less problematic (semantically) than a generic claim such as

(2) Facebook spreads lies

(e.g., Carlson, 1995). (1) describes a particular event that can be located in the timeline (and represented as a string; e.g., Fernando, 2015), while (2) describes multiple events spread over time and arguably some causal complex leading to these events. But are automata suitable causal structures? The hypothesis explored here is that if the granularity is bounded, finite automata provide useful approximations that can be refined and elaborated in various directions.

action language	institution	finite/regular
action signature	$\Sigma \in \mathbf{Sign}$	(A, V)
transition system	$\mathbf{Mod}(\Sigma)$	set of strings
history	Σ-model	string
axiom, query	Σ-sentence	expression

Table 1: Between Gelfond and Lifschitz (1998) and Goguen and Burstall (1992) via Kleene (1956)

Table 1 outlines how this hypothesis is explored below. Bounded granularity is understood as a finite vocabulary (A, V) of acts and fluent-value pairs, collected in a category $\mathbf{Sign}$ of signatures Σ, indexing a set $\mathbf{Mod}(\Sigma)$ of Σ-models and a set $Sen(\Sigma)$ of Σ-sentences in a logical system called an *institution* (Goguen and Burstall, 1992), heading the middle column of Table 1. Exactly what

correspondences to read from Table 1 will take up the entire paper to explain. To reduce disappointment, certain remarks are in order now. The term *action language* in the leftmost column of Table 1 (and throughout this paper) is meant in the narrow sense of Gelfond and Lifschitz (1998); *no* attempt is made to cover all the various approaches that build on labelled transitions (e.g., dynamic logic, process algebras, Markov decision processes and variants). The middle and rightmost columns of Table 1 are connected in Fernando (2022) by notions of

transition signature Σ, Σ-strip and X-projection

which the present work tailors to finite pieces (A, V) of a set Act of acts and a set-valued function Val whose domain, $dom(\mathsf{Val})$, consists of variables that acts can affect.[1] Alongside a certain function

$$\mathsf{af} : \mathsf{Act} \to 2^{dom(\mathsf{Val})}$$

mapping an act e to the set $\mathsf{af}(e)$ of variables that e can affect, the pair $(\mathsf{Act}, \mathsf{Val})$ defines an institution that can be contrasted with Table 1's leftmost column. The function af aids grammatical inference, generalizing

a string *up* to an automaton (language),

rather than particularizing

a transition system *down* to a history.

This shift in perspective (from the left to right columns of Table 1) treats transitions shaped by the preconditions and effects of acts more as a problem to be tackled with strings of refinable granularity, and less as contextual assumptions to be tested by extracting certain histories from them.

We start in section 2 by drawing out action signatures implicit in Kleene (1956) and elsewhere. Linguistic interest in states (e.g., the aspect hypothesis from Dowty (1979, page 71)) motivates the formation in section 3 of strings that trace histories (pairing states with actions). To step beyond any particular string, a category of signatures is fleshed out in section 4, based on a choice of Act and Val, and an institution built around that category with the help of af.

2 Finite-state action signatures

Transitions $q \xrightarrow{a} q'$ that a finite automaton undergoes from state q to q' labeled by the symbol a arise

[1] A companion to this paper, Fernando (2022) provides motivation from natural language inference and knowledge graphs, and also is perhaps best read before section 4 below.

Kleene (1956)	Gelfond and Lifschitz (1998)
inner cell $\mathcal{M}_i$	fluent $\in \mathbf{F}$
input cell $\mathcal{N}_i$	elementary action $\in \mathbf{E}$

Table 2: Sub-symbolic and symbolic

in Kleene (1956) from k *input cells* $\mathcal{N}_1, \ldots, \mathcal{N}_k$ and m *inner cells* $\mathcal{M}_1, \ldots, \mathcal{M}_m$ as follows

(i) a symbol $a = (b_1, \ldots, b_k)$ specifies a value b_i for each input cell $\mathcal{N}_i$

(ii) a state $q = (v_1, \ldots, v_m)$ specifies a value v_i for each inner cell $\mathcal{M}_i$

(iii) a transition relation $\xrightarrow{a}$ combines m simpler relations $\xrightarrow{a}_1, \ldots, \xrightarrow{a}_m$

$$(v_1, \ldots, v_m) \xrightarrow{a} (v'_1, \ldots, v'_m) \iff (v_1, \ldots, v_m) \xrightarrow{a}_i v'_i \text{ for } 1 \leq i \leq m$$

where $(v_1, \ldots, v_m) \xrightarrow{a}_i v'_i$ means

values $v_1, \ldots, v_m$ for $\mathcal{M}_1, \ldots, \mathcal{M}_m$ and a for $(\mathcal{N}_1, \ldots, \mathcal{N}_k)$ cause the value of $\mathcal{M}_i$ to become v'_i

in accordance with neural activation laws (involving thesholds and two types of connections, inhibitory and excitatory, in McCulloch and Pitts (1943), or in the case of perceptrons, weights, biases and activation functions).

The parameter m (specifying the number of inner cells) comes in Kleene (1956) with an m-tuple $(s_1, \ldots, s_m)$ indicating the number s_i of values an inner cell $\mathcal{M}_i$ can take. Each input cell $\mathcal{N}_i$ is understood to take one of two values (firing or not), leading to 2^k symbols $(b_1, \ldots, b_k)$ from k input cells, alongside $\prod_{i=1}^{m} s_i$ states $(v_1, \ldots, v_m)$ from the m inner cells. Assuming $s_1 = s_2 = \cdots = s_m = s$, a triple (k, m, s) induces an *action signature*, as defined in Gelfond and Lifschitz (1998), where

(i) an *action* is a subset of the full set of k input cells

(ii) a *fluent* is one of m inner cells that can take one of s many *values*.

To be sure, the two sides of Table 2 are different enterprises. An inner cell $\mathcal{M}_i$ and input cell $\mathcal{N}_i$ are ingredients in a representation of a McCulloch-Pitts neural net; a fluent such as *dead*(*adolf*) and action such as *kill*(*adolf*) belong to "formal models of parts of the natural language that are used

for talking about the effects of actions" (Gelfond and Lifschitz, 1998). The subsymbolic|symbolic divide here is real; each side has its own concerns and methods. But to understand (if not reconcile) their differences, it is helpful to consider what they have in common. It is noteworthy that the finite-state developments considered below fall squarely within the logical tradition, involving bits of abstract model theory served up in Gelfond and Lifschitz (1998) under the notion of an action signature.

The distribution of a transition $(v_1, \ldots, v_m) \xrightarrow{a} (v'_1, \ldots, v'_m)$ between m transitions

$$(v_1, \ldots, v_m) \xrightarrow{a}_i v'_i \quad \text{for } 1 \leq i \leq m$$

in Kleene (1956) can be sharpened to leave out any inner cell or input cell that does not feed into the inner cell $\mathcal{M}_i$ described by $\xrightarrow{a}_i$. (There is considerable interest in neural networks that are not fully recurrent.) Such niceties are brushed aside in the "black box" perspective adopted in Rabin and Scott (1959), where

> The internal workings of an automaton will not be analyzed too deeply ... The definition of the internal structure must be general enough to cover all conceivable machines, but it need not involve itself with problems of circuitry. ... we need not consider all the intermediate states that the machine passes through but only those directly preceding the reading of a symbol. [pp 115–116]

Nor is a symbol assumed in Rabin and Scott (1959) to have any of the structure that it has in Kleene (1956), as a set of firing input cells.

A symbol a labelling a transition $\xrightarrow{a}$ appears also as a set in Gelfond and Lifschitz (1998), where it is a subset $a \subseteq \mathbf{E}$ of the set $\mathbf{E}$ of *elementary actions*, and in the *S-languages* of Durand and Schwer (2008), which consist of strings of non-empty sets. Interval relations from Allen (1983) widely used for temporal annotations of events (Pustejovsky et al., 2010) can be represented in Durand and Schwer (2008) as strings that mark out the left and right borders, l_i and r_i, of an interval i. For instance, the relation *meet*(i, i') of an interval i abutting i' is representable as the string

$$\boxed{l_i}\boxed{r_i, l_{i'}}\boxed{r_{i'}} \tag{1}$$

consisting of 3 symbols, each boxed (rather than enclosed in braces $\{,\}$) to reinforce the comic strip reading of the string. As an elementary action, l_i corresponds to BECOME(i) in Dowty (1979) or in Pustejovsky (1991) to a transition from not-i to i. Rather than burying an interval i in points l_i and r_i in (1), we can bring i out as a fluent that can go into a box in

$$\boxed{}\boxed{i}\boxed{i'}\boxed{} \tag{2}$$

for a proper treatment of intervals i, i' (rather than points) as "primitive" (Allen, 1983, page 834).[2] The significance of fluents in Knowledge Representation has been recognized since John McCarthy borrowed the term from Newton in the early days of AI. But as aspects of states that are regarded as internal in both Kleene (1956) and Rabin and Scott (1959), fluents do not ordinarily appear in the strings accepted by a finite automaton unless a special effort is made to ensure that they do so. Such an effort is described in the next section, on the understanding that fluents are the "stative predicates" on which Dowty (1979), among others, builds his aspectual calculus (Fernando, 2015, 2019).

3 From histories to strings

Given a set $\mathbf{A}$ of *actions* a labelling transitions $\xrightarrow{a} \subseteq Q \times Q$ between states in Q, a *history* is a sequence $q_0, a_1, q_1, a_2, \ldots, a_n, q_n$ such that

$$q_{i-1} \xrightarrow{a_i} q_i \quad \text{for } 1 \leq i \leq n$$

(Gelfond and Lifschitz, 1998). From the previous section, string (1), $\boxed{l_i}\boxed{r_i, l_{i'}}\boxed{r_{i'}}$, focuses on the action labels in the transitions

$$q_0 \xrightarrow{\boxed{l_i}} q_1 \xrightarrow{\boxed{r_i, l_{i'}}} q_2 \xrightarrow{\boxed{r_{i'}}} q_3$$

while (2), $\boxed{}\boxed{i}\boxed{i'}\boxed{}$, focuses on the states $q0, q1, q2, q3$, saying the (propositional) fluent i holds at q_1, and fluent i' at q_2. Let us refer to the underlying action signature here as

Example S. For the S-language representation of Allen interval relations between i and i', let $\mathbf{F} = \{i, i'\}$, $\mathbf{E} = \{l_i, r_i, l_{i'}, r_{i'}\}$ (with $\mathbf{A} \subseteq 2^{\mathbf{E}}$), $\mathbf{V} = \{1, 0\}$ (for propositional fluents).

Next, let us see how $\boxed{l_i}\boxed{r_i, l_{i'}}\boxed{r_{i'}}$ and $\boxed{}\boxed{i}\boxed{i'}\boxed{}$ emerge naturally from the string

$$\boxed{l_i}\boxed{i, r_i, l_{i'}}\boxed{i', r_{i'}}\boxed{} \tag{3}$$

[2] Strings such as (2) are applied to temporal annotation in Woods and Fernando (2018), and may have more than one fluent in a box, as in the representation $\boxed{}\boxed{i, i'}\boxed{}$ of *equal*(i, i').

with the help of a few definitions. Given a string $s = \alpha_1 \cdots \alpha_n$ of sets α_i and a set X, let

(i) the *vocabulary* $voc(s)$ *of* s be $\bigcup_{i=1}^{n} \alpha_i$, and

(ii) the X*-reduct* $\rho_X(s)$ *of* s be s intersected componentwise with X

$$\rho_X(\alpha_1 \cdots \alpha_n) = (\alpha_1 \cap X) \cdots (\alpha_n \cap X).$$

For example, if s is $\boxed{l_i}\boxed{r_i, l_{i'}}\boxed{r_{i'}}$ then

$$voc(s) = \{l_i, r_i, l_{i'}, r_{i'}\} \text{ and } \rho_{\{r_i\}}(s) = \boxed{}\boxed{r_i}\boxed{}.$$

Although it clashes with conventions in the action language literature, the use of the term "reduct" here follows established model-theoretic practice, which applies to strings, understood as models of predicate logic with a binary relation symbol $\dot{<}$ on string positions, and unary relation symbols P_u specifying the contents of string positions. More precisely, a string $s = \alpha_1 \cdots \alpha_n$ of subsets α_i of a set U is construed as the U-model $M_U(s)$ with domain/universe $D = \{1, 2, \ldots, n\}$ interpreting $\dot{<}$ as $<$ restricted to D, and P_u as the set of $i \in D$ such that $u \in \alpha_i$, with the effect that

$$P_u(i) \text{ says: } u \text{ occurs at string position } i$$

for every $u \in U$.[3] Under this construal, the X-model $M_X(\rho_X(s))$-model is the U-model $M_U(s)$ with P_u restricted to $u \in X$ (making it the $(\{\dot{<}\} \cup \{P_u\}_{u \in X})$-reduct in the usual model-theoretic sense), and regular languages over the alphabet 2^X are the sets of strings definable in MSO_X, *Monadic Second-Order Logic* with unary predicates P_u from $u \in X$ (a slight variant of a fundamental theorem due to Büchi, Elgot and Trakhtenbrot; e.g., Libkin, 2004, Theorem 7.21).

A reduct $\rho_X(s)$ of s keeps the length of s, leaving the domain of the corresponding model as is. But as each $u \in voc(s) \setminus X$ is dropped from $voc(\rho_X(s))$, it is not surprising that compressing $\rho_X(s)$ may be in order. This is the case in Durand and Schwer (2008), where an S*-word* is a string of non-empty sets. To ensure that the X*-projection* $d_X(s)$ *of* an S-word s is still an S-word, every occurrence of $\emptyset$ (appearing as $\square$ *qua* symbol) is deleted from $\rho_X(s)$ to form $d_X(s)$

$$d_X(s) \;:=\; d^{\square}(\rho_X(s)) \;\text{ where } d^{\square} \text{ deletes } \square.$$

[3]Forming unary predicates P_u from elements u of a symbol α is "unconventional" (Vu et al., 2018), the custom being to name unary predicates P_α after the string alphabet's symbols α in their entirety. This shift from actions a to elementary actions u is consequential.

For example,

$$d_{\{r_i\}}(\boxed{l_i}\boxed{r_i, l_{i'}}\boxed{r_{i'}}) = d^{\square}(\boxed{}\boxed{r_i}\boxed{}) = \boxed{r_i}.$$

But what about the representation of *meet*(i, i') as $\boxed{}\boxed{i}\boxed{i'}\boxed{}$, which is not even an S-word?

It helps to understand what the deletion $d^{\square}$ of $\emptyset$ does to the states at either side of $\xrightarrow{\emptyset}$. Towards this end, let us fix a function $\mathsf{af} : \mathbf{E} \to 2^{\mathbf{F}}$ specifying the set $\mathsf{af}(e) \subseteq \mathbf{F}$ of fluents that can be *af*fected by an elementary action e. The obvious choice for af in Example S is

$$\mathsf{af}(l_f) = \mathsf{af}(r_f) = \{f\} \;\text{ for every } f \in \mathbf{F}$$

as l_f and r_f designate the left and right borders of f. In general, however, if we know nothing about an elementary action e, we may set $\mathsf{af}(e) = \mathbf{F}$ to leave open the possibility that e *can* affect any fluent.[4] What effect an elementary action can have on a fluent is given by

(i) transitions $\xrightarrow{a}$ labelled by actions a, and

(ii) a function $\mathcal{V} : \mathbf{F} \times Q \to \mathbf{V}$ that specifies the value $\mathcal{V}(f, q)$ of a fluent f at state q, picturing a state q as the set

$$\mathcal{V}_q \;:=\; \{(f, \mathcal{V}(f, q)) \mid f \in \mathbf{F}\}$$

of fluent-value pairs $(f, \mathcal{V}(f, q))$.

Now, insofar as an action $a \subseteq \mathbf{E}$ can affect only the fluents in $\bigcup_{e \in a} \mathsf{af}(e)$, we can expect that

$$(\dagger) \quad \text{whenever } q \xrightarrow{a} q' \text{ and } f \in \mathbf{F} \setminus \bigcup_{e \in a} \mathsf{af}(e), \quad \mathcal{V}(f, q) = \mathcal{V}(f, q').$$

$(\dagger)$ says that for $q \xrightarrow{a} q'$, any difference between $\mathcal{V}_q$ and $\mathcal{V}_{q'}$ can be confined to $\bigcup_{e \in a} \mathsf{af}(e)$. So if $\bigcup_{a \in e} \mathsf{af}(e) = \mathbf{F}$, $(\dagger)$ is vacuously true. On the other hand, for $a = \emptyset$,

$$\bigcup_{e \in \emptyset} \mathsf{af}(e) = \emptyset$$

and $(\dagger)$ reduces to

$$\mathcal{V}_q = \mathcal{V}_{q'} \text{ whenever } q \xrightarrow{\emptyset} q'.$$

[4]In Kleene (1956) where

$$\mathbf{E} = \{\mathcal{N}_1, \ldots, \mathcal{N}_k\} \;\text{ and }\; \mathbf{F} = \{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$$

notice that $\mathcal{M}_i$ need not be put into $\mathsf{af}(\mathcal{N}_j)$ if $\mathcal{N}_j$ is not among the neural net's inputs into $\mathcal{M}_i$.

Thus, if all we can observe about a state q is its set $\mathcal{V}_q$ of fluent-value pairs, then deleting $q \xrightarrow{\emptyset} q'$ keeps us from seeing $\mathcal{V}_q$ a second time as $\mathcal{V}_{q'}$. In other words, the compression $d^{\Box}(s)$ of a string s of actions corresponds on the fluent side to *block compression* $\mathbb{bc}$, defined by induction on the length of a string by

$$\mathbb{bc}(s) := s \qquad \text{if length}(s) < 2$$
$$\mathbb{bc}(\alpha\alpha' s) := \begin{cases} \mathbb{bc}(\alpha' s) & \text{if } \alpha = \alpha' \\ \alpha\mathbb{bc}(\alpha' s) & \text{otherwise} \end{cases}$$

(Fernando, 2015, §3.1). For example,

$$\mathbb{bc}(\rho_{\{i\}}(\boxed{\ }\boxed{i}\boxed{i'}\boxed{\ })) = \mathbb{bc}(\boxed{\ }\boxed{i}\boxed{\ }\boxed{\ }) = \boxed{\ }\boxed{i}\boxed{\ }.$$

In general, $\mathbb{bc}$ removes stutters $\alpha\alpha$, just as $d^{\Box}$ removes $\Box$.

So far, we have focused on propositional fluents, with value set $\mathbf{V} = \{1, 0\}$. A more refined analysis of Example S that extends readily to any finite number of intervals (beyond i, i') can be given with more than two values. To prepare the ground for this, let us define a string $h^{\mathcal{V}}$ of sets from a history h and a function $\mathcal{V} : \mathbf{F} \times Q \to \mathbf{V}$ as follows. Given a history

$$h = q_0, a_1, q_1, a_2, \ldots, a_n, q_n$$

let $h^{\mathcal{V}}$ be the string $\alpha_0\alpha_1 \cdots \alpha_n$ of $n+1$ sets,

$$\alpha_i := \mathcal{V}_{q_i} \cup a_{i+1} \text{ for } 0 \leq i < n$$

and $\alpha_n := \mathcal{V}_{q_n}$. For the particular history h from the transitions

$$q_0 \xrightarrow{\boxed{l_i}} q_1 \xrightarrow{\boxed{r_i, l_{i'}}} q_2 \xrightarrow{\boxed{r_{i'}}} q_3$$

and the function $\mathcal{V}$ described by $\boxed{\ }\boxed{i}\boxed{i'}\boxed{\ }$ (mapping (i, q_1) and (i', q_2) to 1 and all other relevant fluent-state pairs to 0), the string $h^{\mathcal{V}}$ is $\alpha_0\alpha_1\alpha_2\alpha_3$ where

$$\alpha_0\alpha_1 = \boxed{(i,0),(i',0),l_i}\boxed{(i,1),(i',0),r_i,l_{i'}}$$
$$\alpha_2\alpha_3 = \boxed{(i,0),(i',1),r_{i'}}\boxed{(i,0),(i',0)}$$

abbreviated in (3) to $\boxed{l_i}\boxed{i, r_i, l_{i'}}\boxed{i', r_{i'}}\boxed{\ }$ by shortening any fluent-value pair $(f, 1)$ to f, and leaving out any fluent-value pair $(f, 0)$. Now for the aforementioned refinement of Example S, let us work with the new set

$$\mathbf{V}' = \{\text{a, u, d}\}$$

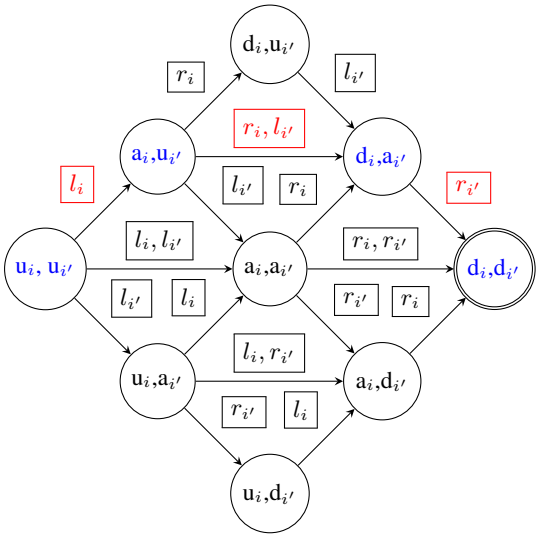

Figure 1: Allen interval relations between i and i'

of values: a for *a*live, u for *u*nborn and d for *d*ead. The point is to divide falsity 0 between u and d, which we reinforce by rewriting 1 to a. This tweak on Example S makes a state q equal to $\mathcal{V}_q$, as illustrated in Figure 1, which for $f \in \{i, i'\}$, attaches

(i) u_f to a state where the value of f is u, saying f is unborn

(ii) a_f to a state where the value of f is a, saying f is alive, and

(iii) d_f to a state where the value of f is d, saying f is dead.

If the fluent value pair (f, v) is written v_f, the blue states and red actions in Figure 1 pick out a history h with $h^{\mathcal{V}}$ equal to

$$\boxed{\text{u}_i,\text{u}_{i'}, l_i}\boxed{\text{a}_i,\text{u}_i, r_i, l_{i'}}\boxed{\text{d}_i,\text{a}_{i'}, r_{i'}}\boxed{\text{d}_i,\text{d}_{i'}}$$

corresponding to (3), $\boxed{l_i}\boxed{i, r_i, l_{i'}}\boxed{i', r_{i'}}\boxed{\ }$. We can capture Russell-Wiener event structures (Kamp and Reyle, 1993, pages 667–674) by generalizing from $\mathbf{F} = \{i, i'\}$ to any set $\mathbf{F}$, keeping

$$\mathsf{af}(l_f) = \mathsf{af}(r_f) = \{f\} \text{ for every } f \in \mathbf{F}.$$

It is convenient here to reformulate functions from $\mathbf{F}$ to {u,a,d} as triples (U, A, D) of subsets of $\mathbf{F}$ with $U \cap A = \emptyset$ and $D = \mathbf{F} \setminus (U \cup A)$. We step from the all-unborn state $(\mathbf{F}, \emptyset, \emptyset)$ to the all-dead state $(\emptyset, \emptyset, \mathbf{F})$ by transitions

$$(U, A, D) \rightsquigarrow_{\mathbf{F}} (U', A', D')$$

which hold precisely if

$$U' \subseteq U \text{ and } A \neq A' \text{ and } D \subseteq D' \subseteq A \cup D$$

as prescribed by the action

$$\{l_f \mid f \in U \setminus U'\} \cup \{r_f \mid f \in D' \setminus D\} \quad (4)$$

which is non-empty (since $A \neq A'$). Labeling $\rightsquigarrow_{\mathbf{F}}$ with the action (4) yields Figure 1 for $\mathbf{F} = \{i, i'\}$.

4 Signature refinements and elaborations

Having represented histories in the previous section by strings formed from material supplied by an action signature, we now modify the notion of an action signature slightly to form strings without relying on histories or transition systems. Signalling this shift, we shorten *elementary actions* to *acts*, and speak of *variables* instead of *fluents*. Each variable x comes with a set $\mathsf{Val}(x)$ of values over which x can range (making the domain of Val the set of variables), and every act e comes with a set $\mathsf{af}(e) \subseteq dom(\mathsf{Val})$ of variables that e can affect. We assume all acts are drawn from some set Act and that

$(*)$ *no* pair (x, v) is in Act with $x \in dom(\mathsf{Val})$.

$(*)$ prevents any act from being confused with a variable instantiation when forming strings (as in the previous section) from sets of acts and variable instantiations.

To bound the granularity of a signature, let us define a *fin-blurring of* Val to be a function V whose domain is a finite subset of Val, mapping each variable x in its domain to a partition $V(x)$ of $\mathsf{Val}(x)$ into finitely many equivalence classes, each smaller than $\mathsf{Val}(x)$. Rather than allowing a completely undifferentiated $V(x) = \{\mathsf{Val}(x)\}$, we can leave x out of $dom(V)$ and insist that there be more than one equivalence class in each partition specified by V.

Definition 1. An $(\mathsf{Act}, \mathsf{Val})$-*signature* is a pair (A, V) of a finite subset A of Act and a fin-blurring V of Val.

The bounded granularity of an $(\mathsf{Act}, \mathsf{Val})$-signature (A, V) rests on the finiteness required of

$$A, \ dom(V), \text{ and } V(x), \text{ for } x \in dom(X).$$

We say a fin-blurring V is *refined by* a fin-blurring V', and write $V \preceq V'$, if V' is defined wherever V is

$$dom(V) \subseteq dom(V')$$

and the partition $V'(x)$ refines $V(x)$

$$(\forall v' \in V'(x))(\exists v \in V(x))\ v' \subseteq v$$

for each $x \in dom(V)$. The partial order $\preceq$ lifts to $(\mathsf{Act}, \mathsf{Val})$-signatures (A, V) and (A', V') in the obvious way

$$(A, V) \preceq (A', V') \iff A \subseteq A' \text{ and } V \preceq V'.$$

The category $\mathbf{Sign}_{\mathsf{Act},\mathsf{Val}}$ of $(\mathsf{Act}, \mathsf{Val})$-signatures is just the set of $(\mathsf{Act}, \mathsf{Val})$-signatures, partially ordered by $\preceq$.

Next, to elaborate on signature refinements, we describe the semantic functor $\mathbf{Mod}$ that is contravariant on $\mathbf{Sign}_{\mathsf{Act},\mathsf{Val}}$. Given a fin-blurring V of Val, a V-*instantiation* is a function g with the same domain as V, picking out, for each $x \in dom(V)$, some equivalence class $g(x) \in V(x)$.[5] Bringing in a finite subset A of Act, we define an (A, V)-*box* to be the union $a \cup g$ of a subset a of A and a V-instantiation g. (This union is disjoint by $(*)$.) An (A, V)-model will be a string of (A, V)-boxes constrained by the function $\mathsf{af} : \mathsf{Act} \to 2^{dom(\mathsf{Val})}$ specifying the set $\mathsf{af}(e)$ of variables that an act e can affect. A variable is said to be *unaffected by* a set $a \subseteq \mathsf{Act}$ of acts if it belongs to the set

$$\overline{\mathsf{af}}(a) := dom(\mathsf{Val}) \setminus \bigcup_{e \in a} \mathsf{af}(e)$$

of variables that *no* act in a affects. For example,

$$\overline{\mathsf{af}}(\emptyset) = dom(\mathsf{Val})$$

as there is no act in $\emptyset$ to affect a variable. Given an (A, V)-box α and a variable $x \in dom(\alpha)$, let us agree that the *instantiation of* x *at* α is $g(x)$ where g is the set difference $\alpha \setminus A$ (recovering the V-instantiation from the disjoint union). We say α and α' are x-*equivalent* and write $\alpha =_x \alpha'$ if x has

[5] Allowing the set of values to vary with the variable taking these values leads to records; stepping to equivalence classes of values leads to record types (e.g. Cooper and Ginzburg, 2015). Support for the move here to types can be found in the following remarks by a leading figure in action languages

> I used to start representation of knowledge relevant to the problem with asking: *What are the objects of our domain?* After some experience I came to (now obvious) realization that the question is wrong. It works for simple problems but does not lead to general and elaboration tolerant solutions. The right question is *What are the SORTS of objects relevant to the domain and what is the relationship between these sorts*?

(Gelfond, 2015, slide 6 of 40).

the same instantiation at α and α'. To give af some bite, we require that adjacent boxes in a string are x-equivalent unless the boxes are linked by an act affecting x.

Definition 2. Given an $(\mathsf{Act}, \mathsf{Val})$-signature (A, V), an (A, V)-*strip* is a string $\alpha_1 \cdots \alpha_n$ of (A, V)-boxes α_i such that $\alpha_n \cap A = \emptyset$ and for all i such that $1 \leq i < n$, $\alpha_i \cap A \neq \emptyset$ and

$$\alpha_i =_x \alpha_{i+1} \text{ for each } x \in \overline{\mathsf{af}}(\alpha_i \cap A).$$

Given $(A, V) \preceq (A', V')$ in $\mathbf{Sign}_{\mathsf{Act},\mathsf{Val}}$, we project a string of (A', V')-boxes to a string of (A, V)-boxes in two steps.

STEP 1: The (A, V)-*coercion of* a string $\alpha'_1 \cdots \alpha'_n$ of (A', V')-boxes α'_i is the componentwise (A, V)-coercions of its (A', V')-boxes α'_i

$$(\alpha'_1 \cdots \alpha'_n)_{A,V} := (\alpha'_1)_{A,V} \cdots (\alpha'_n)_{A,V}$$

where the (A, V)-*coercion* of an (A', V')-box $a' \cup g'$ is the (A, V)-box

$$(a' \cup g')_{A,V} := (a' \cap A) \cup g'_V$$

formed from the union of a' intersected with A and the V-instantiation g'_V mapping $x \in dom(V)$ to the unique $V(x)$-equivalence class blurring $g'(x)$

$$g'_V(x) := \text{unique } c \in V(x) \text{ such that } g'(x) \subseteq c$$

(which is well-defined since g' is a V'-instantiation and $V \preceq V'$).

STEP 2**:** The (A, V)-*compression of* a string s of (A, V)-boxes is the string

$$\gamma_{A,V}(s) := \begin{cases} d^{\Box}(s) & \text{if } V = \emptyset \\ \gamma^A(s) & \text{otherwise} \end{cases}$$

obtained from s by deleting either all occurrences of the empty box $\Box$ in s, in case V is empty, or else all stutters $\alpha\alpha$ disjoint from A

$$\gamma^A(s) := s \qquad \text{if length}(s) < 2$$

$$\gamma^A(\alpha\alpha' s) := \begin{cases} \gamma^A(\alpha' s) & \text{if } \alpha = \alpha' \text{ and} \\ & \alpha \cap A = \emptyset \\ \alpha\, \gamma^A(\alpha' s) & \text{otherwise} \end{cases}$$

making, for example, $\boldsymbol{bc} = \gamma^{\emptyset}$.

Definition 3. Given $(A, V) \preceq (A', V')$ and a string s' of (A', V')-boxes, the (A, V)-*projection of* s', written $\kappa_{A,V}(s')$, is the (A, V)-compression of the (A, V)-coercion of s'

$$\kappa_{A,V}(s') := \gamma_{A,V}({s'}_{A,V}).$$

Definition 4. An (A, V)-*model* is the (A, V)-projection of some (A', V')-strip, where $(A, V) \preceq (A', V')$.

To understand the (A, V)-models from Definition 4 as models in the usual model-theoretic sense, we can start by reformulating the notion of an (A, V)-projection in terms of the translation scheme machinery in, for example, Makowsky (2004). More notation is useful here. A V-*pairing* (x, c) is an element of the set

$$\sum V := \{(x, c) \mid x \in dom(V) \text{ and } c \in V(x)\}$$

(from which V-instantiations are formed). The *vocabulary* $voc(A, V)$ *of* an $(\mathsf{Act}, \mathsf{Val})$-signature (A, V) is the union

$$voc(A, V) = A \cup \sum V$$

of acts in A and V-pairings (again disjoint, by $(*)$). Step 1, the (A, V)-coercion, reduces the $\mathrm{MSO}_{\{u\}}$-formula $P_u(x)$, for every V-pairing u, to the $\mathrm{MSO}_{\sum V'}$-formula

$$P_u^{V,V'}(x) := \bigvee\{P_{u'}(x) \mid u' \in f_{V,V'}(u)\}$$

formed from the set

$$f_{V,V'}(x, c) = \{(x, c') \mid c' \subseteq c \text{ and } c' \in V'(x)\}$$

of V'-pairings that refine the V-pairing $(x, c) = u$ (reversing the step from V'-pairings to V-pairings in g'_V). For Step 2, let us form the $\mathrm{MSO}_{\sum V'}$-formula

$$\psi_u^{V,V'}(x) := P_u^{V,V'}(x) \wedge \neg\exists y(xSy \wedge P_u^{V,V'}(y))$$

saying u occurs at x (under the translation $P_u^{V,V'}$ of P_u into V'-pairings) but not at its successor, where S is the usual successor relation of $\dot{<}$

$$xSy := x\dot{<}y \wedge \neg\exists z(x\dot{<}z \wedge z\dot{<}y).$$

The (A, V)-compression of the (A, V)-coercion of a string s' of (A', V')-boxes restricts the domain/universe of $M_{voc(A,V)}(s')$ to string positions x satisfying the disjunction

$$\phi_{A,V,V'}(x) := \chi_A(x) \vee \chi'_{V,V'}(x) \vee \exists y(xSy \wedge \chi_A(y))$$

where $\chi_A(x)$ says an act in A is done at x

$$\chi_A(x) := \bigvee\{P_e(x) \mid e \in A\}$$

(corresponding to $\Box$-removal $d^{\Box}$) while $\chi'_{V,V'}(x)$ says some V-pairing holds at x (under the V'-translations $P_u^{V,V'}$) but not at its successor

$$\chi'_{V,V'}(x) \;:=\; \bigvee\{\psi_u^{V,V'}(x) \mid u \in \sum V\}$$

(corresponding to stutter-removal $\not bx$). It is convenient here that $\dot{<}$, rather than S, is primitive, as $\kappa_{A,V}(s)$ simply restricts $\dot{<}$ to string positions satisfying $\phi_{A,V,V'}$, and similarly with P_u as $P_u^{V,V'}$, for every V-pairing u. Inasmuch as a string position x represents time, and an act in A represents a force for changing a V-pairing, the restriction to the disjunction $\phi_{A,V,V'}(x)$ says there is no time (discernible in (A,V)) without force (in A) or without change (in a V-pairing or, as expressed by the third disjunct of $\phi_{A,V,V'}(x)$, an act at the next box).

While the (A,V)-projection of an (A,V)-strip does not alter the (A,V)-strip, the (A,V)-projection of an (A',V')-strip may fail to be an (A,V)-strip if (A',V') differs from (A,V). Given an (A,V)-model s, if A' is a minimal subset of Act such that s is the (A,V)-projection of an (A',V')-strip, then the set difference $A' \setminus A$ contains assumptions that the function af uses to explain the transitions within s. This suggests that (A,V)-strips are the most favoured (A,V)-models (carrying, as it were, the greatest weight), and that every addition to $A' \setminus A$ makes the (A,V)-projection of an (A',V')-strip less plausible as an (A,V)-model.

That said, $\mathsf{af}(e)$ only says what variables an act e can affect, and falls far short of specifying how V-pairings under e change (or, as the Frame Problem behind action languages spotlights, do *not* change). Four different action description languages (STRIPS, $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$) are defined in Gelfond and Lifschitz (1998) and many more in papers that follow it, describing transition systems from which histories and action query languages are defined, filling out the leftmost column of Table 1. The preconditions and effects of acts that action description languages build into a transition system have yet to go into strings prescribed by Definition 4 as Σ-models, for $\Sigma = (A,V)$. Given the minimal information encoded in af (not to mention the information lost when turning a history h into a string $h^{\mathcal{V}}$[6]), Definition 4 is bound to overgenerate. But this is where the bottom row of Table 1 enters; every signature Σ comes with a set $Sen(\Sigma)$ of Σ-sentences and a relation $\models_\Sigma$ between Σ-models and Σ-sentences, mapping a Σ-sentence φ into the set

$$\mathbf{Mod}_\Sigma(\varphi) \;:=\; \{M \in \mathbf{Mod}(\Sigma) \mid M \models_\Sigma \varphi\}$$

of Σ-models Σ-satisfying φ. For **Sign** given by $(\mathsf{Act},\mathsf{Val})$-signatures (A,V), an (A,V)-sentence can be an $\mathrm{MSO}_{voc(A,V)}$-sentence, but any notational system for regular languages over the alphabet $2^{voc(A,V)}$ (such as Kleene (1956)'s regular expressions) suffices so that whenever $(A,V) \preceq (A',V')$, the inverse image

$$\{s' \in \mathbf{Mod}(A',V') \mid \kappa_{A,V}(s') \in \mathbf{Mod}_{A,V}(\varphi)\}$$

of $\mathbf{Mod}_{A,V}(\varphi)$ under $\kappa_{A,V}$ restricted to (A',V')-models can be expressed by an (A',V')-sentence $\langle A,V\rangle\varphi$ as $\mathbf{Mod}_{(A',V')}(\langle A,V\rangle\varphi)$. The biconditional

$$s' \models_{(A',V')} \langle A,V\rangle\varphi \iff \kappa_{A,V}(s') \models_{(A,V)} \varphi$$

then becomes the *Satisfaction Condition* characteristic of an institution, where Sen maps a refinement $\sigma : (A,V) \preceq (A',V')$ *co*variantly to a function $Sen(\sigma)$ sending φ to $\langle A,V\rangle\varphi$, complementing the *contra*variance of $\mathbf{Mod}(\sigma)(s') = \kappa_{A,V}(s')$. Observe that the finiteness of $voc(A,V)$ is crucial for (A,V)-projections to be computable by a finite-state transducer, making the inverse image above regular, and ensuring the decidability of entailments (from axioms to queries) given by inclusions

$$\mathbf{Mod}_\Sigma(axioms) \subseteq \mathbf{Mod}_\Sigma(queries).$$

(By contrast, recall that inclusions between say, context-free languages is undecidable.)

There is a delicate interplay here between bounding granularity (to keep representations finite-state and entailments decidable) and accommodating the open-endedness of meaning in natural language by a Satisfaction Condition regulating refinements marked by signature morphisms. Beyond the open-ended modifications of events famously pointed out in Davidson (1967) (calling for indefinite additions to A and V alike), there are preconditions and effects of acts to express through Sen. Given the manner in which acts in A combine with V-pairings for (A,V)-boxes, preconditions constrain the alphabet of (A,V)-boxes, while effects constrain string positions and their successors. But constraints over longer distances can also be expressed through Sen. And just as a signature can be varied, so too can an institution.

[6] This is arguably in keeping with the black box perspective of Rabin and Scott (1959).

References

J.F. Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

G.N. Carlson. 1995. Truth conditions of generic sentences: Two contrasting views. In G.N. Carlson and F.J. Pelletier, editors, *The Generic Book*, pages 224–237. University of Chicago Press.

R. Cooper and J. Ginzburg. 2015. TTR for natural language semantics. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantic Theory*, second edition, pages 375–407. Wiley-Blackwell.

D. Davidson. 1967. The logical form of action sentences. In N. Rescher, editor, *The Logic of Decision and Action*, pages 81–95. University of Pittsburgh Press.

D.R. Dowty. 1979. *Word Meaning and Montague Grammar*. Reidel.

I.A. Durand and S.R. Schwer. 2008. A tool for reasoning about qualitative temporal information: the theory of S-languages with a Lisp implementation. *J. Univers. Comput. Sci.*, 14(20):3282–3306.

T. Fernando. 2015. The semantics of tense and aspect: a finite-state perspective. In S. Lappin and C. Fox, editors, *Handbook of Contemporary Semantic Theory*, second edition, pages 203–236. Wiley-Blackwell.

T. Fernando. 2019. Projecting temporal properties, events and actions. In *Proc 13th International Conference on Computational Semantics*, pages 1–12.

T. Fernando. 2022. Strings from neurons to language. In *Proc. Natural Logic meets Machine Learning III*. ESSLLI 2022 workshop, Galway (Ireland).

M. Gelfond. 2015. Modular action language ALM. Slides for a Knowledge Representation seminar available at `http://redwood.cs.ttu.edu/~mgelfond/TALKS/alm.pdf`.

M. Gelfond and V. Lifschitz. 1998. Action languages. *Linköping Electronic Articles in Computer and Information Science*, 3(16).

J.A. Goguen and R.M. Burstall. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM*, 39(1):95–146.

J. Heinz and J. Sempere, editors. 2016. *Topics in Grammatical Inference*. Springer-Verlag.

C. de la Higuera. 2010. *Grammatical Inference: Learning Automata and Grammars*. Cambridge University Press.

D. Inclezan and M. Gelfond. 2016. Modular action language. *Theory and Practice of Logic Programming*, 16:189–235.

H. Kamp and U. Reyle. 1993. *From Discourse to Logic*. Kluwer Academic Publishers.

S.C. Kleene. 1956. Representation of events in nerve nets and finite automata. In C. Shannon and J. McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press.

L. Libkin. 2004. *Elements of Finite Model Theory*. Springer.

J.A. Makowsky. 2004. Algorithmic uses of the Feferman-Vaught theorem. *Annals of Pure and Applied Logic*, 126:159–213.

W. S. McCulloch and W. H. Pitts. 1943. A logical calculus of the ideas immanent in nervous activity. *Bull. Math. Biophys.*, 5:115–133.

J. Pustejovsky. 1991. The syntax of event structure. *Cognition*, 41:47–81.

J. Pustejovsky, K. Lee, H. Bunt, and L. Romary. 2010. ISO-TimeML: An international standard for semantic annotation. In *Proc 7th International Conference on Language Resources and Evaluation (LREC'10)*, pages 394–397.

M.O. Rabin and D.S. Scott. 1959. Finite automata and their decision problems. *IBM Journal of Research and Development*, 3:114–125.

M.H. Vu, A. Zehfroosh, K. Strother-Garcia, M. Sebok, J. Heinz, and H.G. Tanner. 2018. Statistical relational learning with unconventional string models. *Frontiers in Robotics & AI*, 5.

D. Woods and T. Fernando. 2018. Improving string processing for temporal relations. In *Proc 14th Joint ISO-ACL Workshop on Interoperable Semantic Annotation (ISA-14)*, pages 76–86.