23

Finite-state Methods Featuring Semantics

Tim Fernando

23.1 Introduction

"It may turn out to be very useful for semantic representations too." So concludes the abstract of Lauri Karttunen's COLING 84 paper, Features and values (F&V), referring to "the new Texas version of the 'DG (directed graph)' package" which was "primarily intended for representing morphological and syntactic information" (page 28). That the directed graph was essentially a finite automaton may have been too obvious an observation for F&V to state — or, assuming it had already been stated, restate. Be that as it may, this observation is used in Fernando 2016 to extract typed features structures from Robin Cooper's record type approach to frames (Fillmore 1982, Cooper 2012). I restate the observation here to develop its uses for semantic representations further, egged on by the aforementioned statement from F&V, and (as with Kornai 2017) the prospects of bringing together "two lines of research that lie at the opposite ends on the field" (Karttunen 2007). Just *how* to view feature structures as finite automata is detailed in the next section (section 2); *why* this view might pay off is explored in section 3.

As bottom-dwellers in the Chomsky hierarchy, finite automata have well-known limitations to test the maxim *keep it simple*. Finite-state methods are structured below around semantic notions $\models$ of satisfaction between models and sentences, kept simple through Leibniz's Law, Identity of Indiscernibles. A logical formalism well-known from Hen-

Tokens of Meaning: Papers in Honor of Lauri Karttunen.
Cleo Condoravdi and Tracy Holloway King (eds.).
Copyright © 2018, CSLI Publications.

nessy & Milner 1985 (among other papers) is applied to feature structures in section 2 broadly along the lines of Blackburn 1993, but with particular attention to certain sets of strings to which directed graphs can be reduced, called trace sets. A set Σ of attributes is paired with a trace set $T \subseteq \Sigma^*$ for a *signature* (Σ, T), picking out the set $Mod(\Sigma, T)$ of trace sets L sandwiched between T and Σ^*

$$T \subseteq L \subseteq \Sigma^*.$$

A trace set $L \in Mod(\Sigma, T)$ is a (Σ, T)-*model*, as the notation $Mod(\Sigma, T)$ suggests, against which to evaluate a Σ-sentence. A (Σ, T)-model L can be construed as a record of record type (Σ, T) with an s-component L_s, for every string $s \in T$, that is a trace set satisfying a Σ-sentence φ precisely if L satisfies the Σ-sentence $\langle s \rangle \varphi$

$$L \models \langle s \rangle \varphi \quad \iff \quad L_s \models \varphi.$$

To analyze satisfaction $\models$, it suffices to keep the set Σ in a signature (Σ, T) finite, and integrate different signatures within a category (following the so-called Grothendieck construction). Behind the somewhat technical details below is the intuition that signatures are bounded granularities that simplify calculations of satisfaction $\models$. That simplification, called the *Translation Axiom* in Barwise 1974 (page 235) and the *Satisfaction Condition* in Goguen & Burstall 1992 (page 102), applies to unification in F&V with negative and disjunctive constraints that refine the sets $Mod(\Sigma, T)$.

Supposing a typed feature structure can be viewed as a finite automaton (which section 2 takes pains to show), *so what*? To make the view compelling, we turn in section 3 to runs of finite automata with the eventual goal of understanding these runs as uses of linguistic resources encoded by typed feature structures. An approach to temporality in which time arises from running automata is presented paralleling section 2, with Monadic Second Order logic in place of Hennessy-Milner logic, and superposition in place of unification (for building models bottom-up, subject to constraints). Careful attention is paid to shifts in bounded granularity and to the assorted forces that take shape as granularity is refined. This is in contrast to the practice of fixing some space of possible worlds once and for all, without any provisions for varying granularity.

23.2 From Features to Strings and Types

Some notions taken up in F&V are collected in the first column of Table 1, which we analyze in this section according to the second column.

path $\langle a_1 \cdots a_n \rangle$ of attributes	string $a_1 \cdots a_n$
(rooted) directed graph G	set $L(G)$ of strings
generalize(G, G')	$L(G) \cap L(G')$
constraints C	set Φ_C of sentences with $\neg, \vee$
unify$_C(G, G')$	$L(G) \cup L(G')$ if it satisfies Φ_C

Table 1

We take for granted in Table 1 a set Σ of *attributes* $(a, a_i, \ldots)$, and define a Σ-*deterministic system* to be a partial function $\delta : Q \times \Sigma \rightharpoondown Q$ to some set Q of nodes from the set $Q \times \Sigma$ of node-attribute pairs. We picture a triple $(q, a, \delta(q,a))$ in δ as a deterministic transition $q \xrightarrow{a} \delta(q,a)$, and formulate a (rooted) directed graph G as a pair (δ, q) of a Σ-deterministic system δ and a node $q \in Q$. To define the language $L(G) \subseteq \Sigma^*$, let $\delta_q : \Sigma^* \rightharpoondown Q$ be the $\subseteq$-smallest subset F of $\Sigma^* \times Q$ such that

(i) $(\epsilon, q) \in F$, and
(ii) $(sa, q'') \in F$ whenever $(s, q') \in F$ and $(q', a, q'') \in \delta$.

Now, if the directed graph G is the pair (δ, q), then its language $L(G)$ is the set $dom(\delta_q)$ of strings s for which $\delta_q(s)$ is defined. The language $dom(\delta_q)$ is called the *trace set of* (δ, q), and strings in $dom(\delta_q)$ are called *traces of* (δ, q).

Next, with Σ fixed, we express constraints through the set $sen(\Sigma)$ of sentences φ generated from attributes $a \in \Sigma$ by the grammar

$$\varphi \quad ::= \quad \top \mid \langle a \rangle \varphi \mid \neg\varphi \mid \varphi \vee \varphi'$$

interpreted against a Σ-deterministic system $\delta : Q \times \Sigma \rightharpoondown Q$ by a binary relation $\models^\delta \subseteq Q \times sen(\Sigma)$ that treats $\top$ as a tautology

$$q \models^\delta \top \quad \text{for every } q \in Q,$$

$\langle a \rangle$ as the Diamond modal operator with accessibility relation $\delta(\cdot, a)$

$$q \models^\delta \langle a \rangle \varphi \iff (q,a) \in dom(\delta) \text{ and } \delta(q,a) \models^\delta \varphi,$$

$\neg$ as Boolean negation

$$q \models^\delta \neg\varphi \iff \text{not } q \models^\delta \varphi$$

and $\vee$ as Boolean disjunction

$$q \models^\delta \varphi \vee \varphi' \iff q \models^\delta \varphi \text{ or } q \models^\delta \varphi'$$

(Hennessy & Milner 1985, Blackburn 1993). Collecting the sentences in $sen(\Sigma)$ that q $\models^\delta$-satisfies in

$$sen_\Sigma(\delta, q) \ := \ \{\varphi \in sen(\Sigma) \mid q \models^\delta \varphi\},$$

it turns out that directed graphs satisfy the same subset of $sen(\Sigma)$ precisely if they have the same trace set

$$sen_\Sigma(\delta, q) = sen_\Sigma(\delta', q') \iff dom(\delta_q) = dom(\delta'_{q'}) \qquad (23.28)$$

(Hennessy & Milner 1985).[1] Under Leibniz's Identity of Indiscernibles, with discernibility based on $sen(\Sigma)$, (23.1) reduces a directed graph (δ, q) to its trace set $dom(\delta_q)$. The trace set captures a fragment of $sen(\Sigma)$

$$dom(\delta_q) \ = \ \{s \in \Sigma^* \mid \langle s\rangle\top \in sen_\Sigma(\delta, q)\}$$

consisting of sentences of the form $\langle s\rangle\top$, where for every $\varphi \in sen(\Sigma)$, the sentence $\langle s\rangle\varphi$ in $sen(\Sigma)$ is defined by induction on $s \in \Sigma^*$, starting with the null string ϵ,

$$\langle\epsilon\rangle\varphi \ := \ \varphi$$

and using modal operators $\langle a\rangle$ elsewhere

$$\langle as\rangle\varphi \ := \ \langle a\rangle\langle s\rangle\varphi$$

so that $\langle a_1 \cdots a_n\rangle\varphi$ is $\langle a_1\rangle \cdots \langle a_n\rangle\varphi$ and

$$q \models^\delta \langle s\rangle\varphi \iff s \in dom(\delta_q) \text{ and } \delta_q(s) \models^\delta \varphi. \qquad (23.29)$$

In the remainder of this section, we replace q and δ in (23.2) by a prefix-closed language over Σ with derivatives (in §2.1), and expand Σ to flesh out Table 1 (in §2.2), systematized category-theoretically (in §2.3) to link up with section 3.

23.2.1 Languages and Transitions to Derivatives

Given a set $L \subseteq \Sigma^*$ of strings over Σ and a string s over Σ, the *s-derivative of L* is the set

$$L_s \ := \ \{s' \mid ss' \in L\}$$

of strings that put after s belong to L (Brzozowski 1964). For any Σ-deterministic system $\delta : Q \times \Sigma \rightharpoonup Q$ and node q, one can check that $dom(\delta_q)$ is the set of strings s such that the null string ϵ is in the s-derivative of $dom(\delta_q)$

$$s \in dom(\delta_q) \iff \epsilon \in (dom(\delta_q))_s$$

and that for every $s \in dom(\delta_q)$, the s-derivative of $dom(\delta_q)$ is the trace set of $(\delta, \delta_q(s))$

$$(dom(\delta_q))_s = dom(\delta_{q'}) \text{ where } q' = \delta_q(s).$$

[1] Readers familiar with, for example, Barwise & Moss 1996 will note that determinism simplifies matters considerably, reducing bisimulation equivalence between (δ, q) and (δ', q') to trace equivalence $dom(\delta_q) = dom(\delta'_{q'})$, and allowing us to talk of sets of strings instead of non-well-founded sets.

Indeed, the chain of equivalences

$$\begin{aligned} a_1 a_2 \cdots a_n \in L &\iff a_2 \cdots a_n \in L_{a_1} \\ &\iff \cdots \iff \epsilon \in L_{a_1 \cdots a_n} \end{aligned}$$

from $a_1 \cdots a_n$ to the null string ϵ means that L is accepted by the deterministic automaton with

- s-derivatives L_s as states
- initial state $L = L_\epsilon$
- a-transitions from L_s to L_{sa} (for every symbol $a \in \Sigma$)
- final (accepting) states L_s such that $\epsilon \in L_s$.

The s-derivative of L equals the s'-derivative of L precisely if s and s' concatenate with the same strings to produce strings in L

$$L_s = L_{s'} \iff (\forall w \in \Sigma^*)\ (sw \in L \iff s'w \in L)$$

so that the Myhill-Nerode Theorem says that for finite Σ,

$$L \text{ is regular} \iff \{L_s \mid s \in \Sigma^*\} \text{ is finite}$$

(e.g. Hopcroft & Ullman 1979). Note that L_s is non-empty precisely if s is the prefix of some string in L. Moreover, if L_s is empty then so is L_{sa} for every $a \in \Sigma$. That is, $\emptyset$ is a sink state that we may safely exclude from the states of the automaton above, at the cost of making the transition function partial.

Let us call a language L *prefix-closed* if for all $sa \in L$, $s \in L$. Note that trace sets are prefix-closed and non-empty. Let $\mathit{Mod}(\Sigma)$ denote the set

$$\mathit{Mod}(\Sigma) := \{L \subseteq \Sigma^* \mid L \neq \emptyset \text{ and } L \text{ is prefix-closed}\}$$

of non-empty prefix-closed subsets of Σ^*, and let us refer to an element of $\mathit{Mod}(\Sigma)$ as a Σ-*state*. Not only are trace sets Σ-states, but conversely, if $\hat{\delta}$ is the Σ-deterministic system

$$\{(L, a, L_a) \mid L \in \mathit{Mod}(\Sigma) \text{ and } a \in \Sigma \cap L\}$$

then every Σ-state L is the trace set of $(\hat{\delta}, L)$. Keeping $\hat{\delta}$ implicit, a Σ-state L makes an s-transition to its s-derivative L_s precisely if $s \in L$, specializing the biconditional (2) from the previous page to

$$L \models \langle s \rangle \varphi \iff s \in L \text{ and } L_s \models \varphi.$$

23.2.2 Adding Attributes, Types and Constraints

Identity as indiscernibility relative to $\mathit{sen}(\Sigma)$ presupposes that all differences which matter are captured by the set Σ. An obvious problem is that the single trace set $\{\epsilon\}$ cannot differentiate between atomic values. But it is easy enough to introduce for every atomic value v, a fresh attribute a_v to Σ for say, the trace set $\{a_v, \epsilon\}$. At least two objections can be made to this move. The first is that a trace set of $\{\epsilon\}$ is arguably what it means for a value v to be atomic; any larger trace set would make v non-atomic. If "atomic" is understood this way, identity as indiscernibility leaves us no choice but to differentiate between values by making all but perhaps one of them non-atomic. A more serious objection is that if the alphabet Σ is to be finite, then we cannot introduce fresh attributes to Σ indefinitely. Or can we? Given any set $\mathcal{A}$, no matter how large, we can form its set $Fin(\mathcal{A})$ of finite subsets

$$Fin(\mathcal{A}) \ := \ \{\Sigma \subseteq \mathcal{A} \mid \Sigma \text{ is finite}\}$$

and let Σ vary over members of $Fin(\mathcal{A})$; each attribute $a \in \mathcal{A} - \Sigma$ added to Σ leads to the different member $\Sigma \cup \{a\}$ of $Fin(\mathcal{A})$. The challenge then becomes to implement the variations in Σ systematically. This is where signatures and institutions enter.

But first, it will prove convenient to expand $sen(\Sigma)$ with a modal operator $\Diamond$ for a sentence $\Diamond\varphi$ equivalent to the disjunction over all $s \in \Sigma^*$ of the sentences $\langle s \rangle \varphi$. More precisely,

$$q \models^{\delta} \Diamond\varphi \iff (\exists s \in \Sigma^*)\ q \models^{\delta} \langle s \rangle \varphi \tag{23.30}$$

for any Σ-deerministic system $\delta : Q \times \Sigma \rightharpoonup Q$ and node $q \in Q$. Incorporating $\Diamond$ into $sen(\Sigma)$ and $sen_{\Sigma}(\delta, q)$ for $sen^{\Diamond}(\Sigma)$ and $sen^{\Diamond}_{\Sigma}(\delta, q)$ respectively, it is not dfficult to verify that trace equivalence remains indiscernibility up to $sen^{\Diamond}(\Sigma)$

$$sen^{\Diamond}_{\Sigma}(\delta, q) = sen^{\Diamond}_{\Sigma}(\delta', q') \iff dom(\delta_q) = dom(\delta'_{q'}).$$

Thus, we can again reduce (δ, q) to its trace set $dom(\delta_q)$ and $\models^{\delta}$ to a binary relation $\models_{\Sigma} \subseteq Mod(\Sigma) \times sen^{\Diamond}(\Sigma)$ between a Σ-state L and a sentence $\varphi \in sen^{\Diamond}(\Sigma)$, simplifying (23.3) to

$$\begin{aligned} L \models_{\Sigma} \Diamond\varphi &\iff (\exists s \in \Sigma^*)\ L \models_{\Sigma} \langle s \rangle \varphi \\ &\iff (\exists s \in L)\ L_s \models_{\Sigma} \varphi \end{aligned}$$

(adding the subscript Σ to prepare for the aforementioned variations). As usual, we let $\Box\varphi$ abbreviate $\neg\Diamond\neg\varphi$ for

$$L \models_{\Sigma} \Box\varphi \iff (\forall s \in L)\ L_s \models_{\Sigma} \varphi$$

alongside the Boolen conventions $\varphi \supset \psi$ for $\psi \vee \neg\varphi$, and $\varphi \wedge \psi$ for $\neg(\neg\varphi \vee \neg\psi)$. Given a subset Φ of $sen^{\Diamond}(\Sigma)$, we say a Σ-state L is a Σ-*model of* Φ, and write $L \models_{\Sigma} \Phi$, if it satisfies every sentence in Φ

$$L \models_{\Sigma} \Phi \iff (\forall \varphi \in \Phi)\ L \models_{\Sigma} \varphi.$$

Now, to pick out a particular Σ-state through a sentence φ, let $Uniq_{\Sigma}(\varphi)$

be the set

$$Uniq_{\Sigma}(\varphi) \ := \ \{\Diamond(\varphi \wedge \psi) \supset \Box(\varphi \supset \psi) \mid \psi \in sen(\Sigma)\}$$

of implications $\Diamond(\varphi \wedge \psi) \supset \Box(\varphi \supset \psi)$ ensuring that if ψ should ever occur with φ, it always occurs with φ. Since trace equivalence is indiscernibility with respect to $sen(\Sigma)$, it follows that the sentences ψ appearing in $Uniq_{\Sigma}(\varphi)$ can be restricted to those of the form $\langle s\rangle\top$ for $s \in \Sigma^*$ without changing the Σ-models of $Uniq_{\Sigma}(\varphi)$, and that

(†) for any Σ-model L of $Uniq_{\Sigma}(\varphi)$, and $s, s' \in L$,

$$\text{if } L_s \models_{\Sigma} \varphi \text{ and } L_{s'} \models_{\Sigma} \varphi \ \text{ then } \ L_s = L_{s'}.$$

If we are to introduce an attribute a_v to name a particular value v through the sentence $\langle a_v\rangle\top$, then we must restrict our $(\Sigma\cup\{a_v\})$-states to $(\Sigma \cup \{a_v\})$-models of $Uniq_{\Sigma\cup\{a_v\}}(\langle a_v\rangle\top)$. An attribute a might also be introduced to name a type that applies to more than one $(\Sigma \cup \{a\})$-state, implicating a $(\Sigma\cup\{a\})$-state that fails to satisfy some sentence in $Uniq_{\Sigma\cup\{a\}}(\langle a\rangle\top)$. There is a curious twist here on treatments of "identity and mere likeness" (F&V, page 29) and re-entrancy (connected with a feature path s that appears in the present set-up as a subscript in L_s and inside a modal operator in $\langle s\rangle\varphi$). Σ-states L and L' can be distinct only if some sentence in $sen(\Sigma)$ differentiates them (shifting, as it were, the burden of proof from identification to differentiation, and suggesting refinements of identity through expansions of Σ).

Additional attributes may serve purposes other then refining discernibility. For example, they may provide representations of sentences in $sen^{\Diamond}(\Sigma)$ as follows. Given a subset Φ of $sen^{\Diamond}(\Sigma)$, a sentence $\varphi \in sen^{\Diamond}(\Sigma)$, and a string $s \in \Sigma^*$, let us agree that s *(Σ, Φ)-represents* φ if every Σ-model of Φ satisfies

$$\Box(\varphi \equiv \langle s\rangle\top)$$

where $\varphi \equiv \psi$ is $(\varphi \supset \psi) \wedge (\psi \supset \varphi)$, and consequently, for any Σ-state L,

$$L \models_{\Sigma} \Box(\varphi \equiv \psi) \iff (\forall s \in L)(L_s \models_{\Sigma} \varphi \iff L_s \models_{\Sigma} \psi).$$

Because we can build φ with the connectives $\neg$ and $\vee$, we cannot expect there to be a string that $(\Sigma, \emptyset)$-represents φ. But we can always introduce an attribute $a_{\varphi} \notin \Sigma$ and set Φ to $\{\Box(\varphi \equiv \langle a_{\varphi}\rangle\top)\}$ so that a_{φ} $(\Sigma \cup \{a_{\varphi}\}, \Phi)$-represents φ. And we can put together attributes a_{φ} and a_{ψ} that (Σ, Φ)-represent φ and ψ respectively, as Σ-models of Φ satisfy

$$\Box \ ((\varphi \wedge \psi) \ \equiv \ (\langle a_{\varphi}\rangle\top \wedge \langle a_{\psi}\rangle\top)).$$

We can then avoid the addition of $a_{\varphi\wedge\psi}$, provided we generalize our notion of representation to a language $\hat{L} \subseteq \Sigma^*$ as follows. We say $\hat{L}$ (Σ, Φ)-*represents* φ if for every Σ-model L of Φ and $s \in L$,

$$\begin{aligned} L_s \models_\Sigma \varphi \quad &\iff \quad (\forall s' \in \hat{L})\ L_s \models \langle s' \rangle \top \\ &\iff \quad \hat{L} \subseteq L_s \end{aligned} \tag{23.31}$$

Clearly, a string s (Σ, Φ)-represents φ iff the singleton language $\{s\}$ (Σ, Φ)-represents φ. But why should we care about representing sentences by languages?

Table 1 at the beginning of the present section mentions not only directed graphs G and G' but also constraints C. Directed graphs are formulated here as Σ-states (models), and constraints as subsets of $sen^\diamond(\Sigma)$. A Σ-state can be viewed as a token, and a sentence φ in $sen^\diamond(\Sigma)$ as the type

$$Mod_\Sigma(\varphi) \ := \ \{L \in Mod(\Sigma) \mid L \models_\Sigma \varphi\}$$

of Σ-states satisfying φ. A set $\Phi \subseteq sen^\diamond(\Sigma)$ of sentences amounts to the conjunction $\bigwedge \Phi$ specifying the type

$$Mod_\Sigma(\Phi) \ := \ \bigcap_{\varphi \in \Phi} Mod_\Sigma(\varphi)$$

of Σ-states satisfying every sentence in Φ. Inclusion $\subseteq$ between sets of strings over Σ in (23.4) is easily confused with that between sets $Mod_\Sigma(\varphi)$ and $Mod_\Sigma(\psi)$ of such sets

$$Mod_\Sigma(\varphi) \subseteq Mod_\Sigma(\psi) \quad \iff \quad (\forall L \in Mod(\Sigma))\ L \models_\Sigma \varphi \supset \psi$$

signifying an entailment from φ to ψ (and reversing the direction in (23.4) from the less informative $\hat{L}$ to the more informative L_s). Converting a sentence φ to a Σ-state that (Σ, Φ)-represents it requires a set Φ of constraints that we can find in, if necessary, an expansion of Σ. Resorting to Φ as $\{\Box(\varphi \equiv \langle a_\varphi \rangle \top)\}$ with a_φ thrown into Σ is perhaps too easy, shoving all the work over to Φ. But there is surely a role for Φ, since L can only $(\Sigma, \emptyset)$-represent a sentence with the same Σ-models as $\{\langle s \rangle \top \mid s \in L\}$, leaving out many sentences formed with negation $\neg$ and disjunction $\vee$. The models of a sentence φ that a language $(\Sigma, \emptyset)$-represents are closed under inclusion $\subseteq$

$$(\forall L \in Mod_\Sigma(\varphi))(\forall L' \in Mod(\Sigma))\ L \subseteq L' \text{ implies } L' \models_\Sigma \varphi$$

and intersection $\cap$

$$(\forall L \in Mod_\Sigma(\varphi))(\forall L' \in Mod_\Sigma(\varphi))\ L \cap L' \models_\Sigma \varphi.$$

But closure under intersection fails for the negation $\neg\langle a \rangle \top$, and closure under intersection fails for the disjunction $\langle a \rangle \top \vee \langle a' \rangle \top$ (with two different $\subseteq$-minimal models, for $a \neq a'$).

As a binary operation on directed graphs, unification in F&V is defined on Σ-states, and, pace Blackburn 1993, *not* on sentences (in terms of the connective $\wedge$). The constraints determining when two directed graphs are unifiable does, however, bring in $sen^{\Diamond}(\Sigma)$, as does talk of negative and disjunctive features inasmuch as these involve the $sen^{\Diamond}(\Sigma)$-connectives $\neg$ and $\vee$. Evidently, a mix of Σ-states and Σ-sentences is required. Accordingly, let us pair Σ with a language $T \subseteq \Sigma^*$, revising $Mod(\Sigma)$ to

$$Mod(\Sigma, T) \ := \ \{L \in Mod(\Sigma) \mid T \subseteq L\}$$

and $Mod_\Sigma(\Phi)$, for $\Phi \subseteq sen^{\Diamond}(\Sigma)$, to

$$Mod_{\Sigma,T}(\Phi) \ := \ \{L \in Mod_\Sigma(\Phi) \mid T \subseteq L\}.$$

Then relative to constraints Φ, we can analyze the unification of Σ-states L and L' in terms of $Mod_{\Sigma,L\cup L'}(\Phi)$, which may be empty even if neither $Mod_{\Sigma,L}(\Phi)$ nor $Mod_{\Sigma,L'}(\Phi)$ is, accounting for the partiality of unification

$$L \text{ and } L' \text{ are unifiable relative to } \Phi \quad \Longleftrightarrow \quad Mod_{\Sigma,L\cup L'}(\Phi) \neq \emptyset.$$

Negation and disjunction in features may (or may not) require expanding Σ with $a_{\neg\varphi}$ and $a_{\varphi\vee\psi}$, and Φ with constraints

$$\Box(\varphi' \equiv \langle a_{\varphi'}\rangle\top) \qquad \text{for } \varphi' \in \{\neg\varphi, \varphi \vee \psi\}.$$

Fixing some large set $\mathcal{A}$ to which all the required attributes belong, we let Σ vary over the set $Fin(\mathcal{A})$ of finite subsets of $\mathcal{A}$, and note

Fact 1 *Let $\Sigma' \in Fin(\mathcal{A})$, $\Sigma \subseteq \Sigma'$, $\varphi \in sen(\Sigma)$, and $L' \in Mod(\Sigma')$. Then $L' \cap \Sigma^* \in Mod(\Sigma)$ and*

$$L' \models_{\Sigma'} \varphi \quad \Longleftrightarrow \quad L' \cap \Sigma^* \models_\Sigma \varphi$$

and moreover, for every $s \in L' \cap \Sigma^*$, $(L' \cap \Sigma^*)_s \in Mod(\Sigma)$ and

$$L' \models_{\Sigma'} \langle s\rangle\varphi \quad \Longleftrightarrow \quad (L' \cap \Sigma^*)_s \models_\Sigma \varphi.$$

The first part of Fact 1 says that the attributes that matter in satisfying φ are only those that appear in φ,[2] while the second part interprets the modal operator $\langle s\rangle$ against Σ'-states L' under the presupposition that s belongs to L'.

[2]Fact 1 leaves $\Diamond$ out of φ precisely because $\Diamond$ does not identify the attributes relevant to the satisfaction of sentences built with $\Diamond$. To bring $\Diamond$ into φ in Fact 1, we can add subscripts X ranging over subsets of Σ to make the pertinent attributes in $\Diamond_X$ explicit, with

$$q \models^\delta \Diamond_X\psi \quad \Longleftrightarrow \quad (\exists s \in X^*)\ q \models^\delta \langle s\rangle\psi$$

(Fernando 2016).

23.2.3 The Grothendieck Construction and an Institution

Some category-theoretic structure lurking in Fact 1 will resurface in section 3 under a different guise and is worth spelling out. We fix a large set $\mathcal{A}$ of attributes, and for each finite subset $\Sigma \in \mathit{Fin}(\mathcal{A})$ of $\mathcal{A}$, turn the set $\mathit{Mod}(\Sigma)$ of Σ-states into a category $\mathcal{Q}(\Sigma)$ as follows. A $\mathcal{Q}(\Sigma)$-morphism from Σ-state L to Σ-state L' is a pair (L, s) with $s \in L$ and $L_s = L'$. $\mathcal{Q}(\Sigma)$-morphisms compose by concatenating strings

$$(L, s); (L_s, s') \ := \ (L, ss')$$

and (L, ϵ) is the identity morphism for L. Whenever $\Sigma \subseteq \Sigma' \in \mathit{Fin}(\mathcal{A})$, we define the functor $\mathcal{Q}(\Sigma', \Sigma) : \mathcal{Q}(\Sigma') \to \mathcal{Q}(\Sigma)$ from $\mathcal{Q}(\Sigma')$ to $\mathcal{Q}(\Sigma)$ mapping

- a Σ'-state L' to the Σ-state $L' \cap \Sigma^*$, and
- a $\mathcal{Q}(\Sigma')$-morphism (L', s) to the $\mathcal{Q}(\Sigma)$-morphism $(L' \cap \Sigma^*, \pi_\Sigma(s))$

where $\pi_\Sigma(s)$ is the longest prefix of s in Σ^*

$$\begin{aligned} \pi_\Sigma(\epsilon) \ &:= \ \epsilon \\ \pi_\Sigma(as) \ &:= \ \begin{cases} a\,\pi_\Sigma(s) & \text{if } a \in \Sigma \\ \epsilon & \text{otherwise.} \end{cases} \end{aligned}$$

Construing $\mathit{Fin}(\mathcal{A})$ as a category with morphisms given by inclusion $\subseteq$, the foregoing defines a contravariant functor $\mathcal{Q} : \mathit{Fin}(\mathcal{A})^{op} \to \mathbf{Cat}$ into the category $\mathbf{Cat}$ of small categories. The *Grothendieck construction* (e.g., Tarlecki, Burstall & Goguen 1991) applied to $\mathcal{Q}$ yields the category $\int \mathcal{Q}$ where

- an object is a pair $(\Sigma, L) \in \mathit{Fin}(\mathcal{A}) \times \mathit{Mod}(\Sigma)$, and
- a morphism from (Σ', L') to (Σ, L) is a pair

  $$((\Sigma', \Sigma), (L'', s))$$

  of a $\mathit{Fin}(\mathcal{A})^{op}$-morphism (Σ', Σ) and a $\mathcal{Q}(\Sigma)$-morphism (L'', s) such that

  $$L'' = L' \cap \Sigma^* \text{ and } L = L''_s.$$

Reversing the morphisms in $\int \mathcal{Q}$ for the category $\mathbf{Sign}$ of *signatures* (Σ, L), we define two functors from $\mathbf{Sign}$, one covariant and the other contravariant

(i) $\mathit{sen} : \mathbf{Sign} \to \mathbf{Set}$ with $\mathit{sen}(\Sigma, L) := \mathit{sen}(\Sigma)$ and

$$\mathit{sen}((\Sigma, \Sigma'), (L'', s)) : \ \varphi \mapsto \langle s \rangle \varphi$$

(ii) $\mathit{Mod} : \mathbf{Sign}^{op} \to \mathbf{Cat}$ where the set $\mathit{Mod}(\Sigma, L)$ of Σ-states that $\subseteq$-contain L is turned into a full subcategory of $\mathcal{Q}(\Sigma)$, and

$$\mathit{Mod}((\Sigma', \Sigma), (L'', s)) : \ \hat{L} \mapsto (\hat{L} \cap \Sigma^*)_s.$$

To build an *institution* (Goguen & Burstall 1992) from **Sign**, *sen*, and *Mod*, it remains to form, for every signature (Σ, L), a relation $\models_{\Sigma,L}$ by intersecting $\models_{\Sigma}$ with $Mod(\Sigma, L) \times sen(\Sigma)$. Fact 1 is essentially the *Satisfaction Condition* characterizing institutions

> for every signature (Σ', L'), subset Σ of Σ', string $s \in L' \cap \Sigma^*$, sentence $\varphi \in sen(\Sigma)$, and $\hat{L} \in Mod(\Sigma', L')$,
>
> $$\hat{L} \models_{\Sigma',L'} \langle s \rangle \varphi \quad \Longleftrightarrow \quad (\hat{L} \cap \Sigma^*)_s \models_{\Sigma,L} \varphi$$
>
> where the subscript L above is short for $(L' \cap \Sigma^*)_s$.

Introduced by Goguen and Burstall to cope with the proliferation of logical systems in computer science, the notion of an institution has attracted considerable attention and found numerous applications (e.g. Diaconescu 2012, Kutz et al 2010). Under Fact 1, features and values can be seen as part of that body of work.

23.3 Time for and from Running Automata

It is one thing to encode a linguistic resource as a feature structure equivalent to a finite automaton. It is quite another matter to understand the use of such a resource as the use of a finite automaton. To use a finite automaton is (arguably first and foremost) to run it, accepting strings that end in a final/accepting state. But such runs take place in isolation, whereas it is only in combination with other resources that the encoding or use of a linguistic resource is interesting. The whole point of the category-theoretic approach from the previous section is to relate different feature structures. Similarly, the present section considers runs of an automaton *not* so much in isolation as in combination with other automata, constructing a notion of time from such runs. A simple way to superpose runs of two finite automata is defined in §3.1, and related to the approximation of Priorean temporal models in §3.2 by strings constructed from temporal propositions. We adopt the custom from Artificial Intelligence of referring to temporal propositions as *fluents*. We fix some large set Θ of fluents much as we fixed a large set $\mathcal{A}$ of attributes in the previous section. The plan roughly is to fill out Table 2, embracing Leibniz' Identity of Indiscernibles (as in section 2), with granularity given by a finite subset A of Θ (analogous to $\Sigma \in Fin(\mathcal{A})$ in section 2) to form strings over the alphabet 2^A of subsets of A. The $\subseteq$-larger the subset A, the more refined the A-models and the more expressive the A-sentences can be.

	section 2	section 3
information merge	unify graphs	superpose strings
large set	$\mathcal{A}$ of attributes	Θ of fluents
grain/signature	$\Sigma \in Fin(\mathcal{A})$	$A \in Fin(\Theta)$
model	language over Σ	string over 2^A
sentence	Hennessy-Milner	Monadic Second-Order

Table 2

Helpful examples for orientation are provided by representations of a calendar year at various granularities. The set $A = \{\text{Jan, Feb, \ldots, Dec}\}$ of months suggests the string

$$s_A \ := \ \boxed{\text{Jan}}\boxed{\text{Feb}}\cdots\boxed{\text{Dec}}$$

of length 12. Enlarging A with days d1,d2,. . .,d31

$$A' \ := \ A \cup \{\text{d1,d2\ldots,d31}\}$$

refines s_A to the string

$$s_{A'} \ := \ \boxed{\text{Jan,d1}}\boxed{\text{Jan,d2}}\cdots\boxed{\text{Jan,d31}}\boxed{\text{Feb,d1}}\cdots\boxed{\text{Dec,d31}}$$

of length 366 for a leap year. We draw boxes (instead of the usual curly braces { and }) around sets *qua* symbols to suggest a film strip. A change in A can cause a box to split (much like hairs in Shan 2015), as $\boxed{\text{Jan}}$ in s_A does (30 times) on adding days

$$\boxed{\text{Jan}} \ \rightsquigarrow \ \boxed{\text{Jan,d1}}\boxed{\text{Jan,d2}}\cdots\boxed{\text{Jan,d31}}$$

in $s_{A'}$. Similarly, a common Reichenbachian account of the progressive puts a *reference time* R inside the event time E, splitting $\boxed{\text{E}}$ into 3 boxes

$$\boxed{\text{E}} \ \rightsquigarrow \ \boxed{\text{E}}\boxed{\text{E,R}}\boxed{\text{E}}$$

(one before, one simultaneous, and one after R). This and many other examples in tense and aspect are taken up at length in Fernando 2015. The aim of the present section is to link that work with the previous section through the notion of an institution. The hope is that this might contribute to understanding the use of linguistic resources encoded as feature structures in terms of runs of finite automata — runs that give rise to time at bounded granularities.

23.3.1 From Superposition to Reducts and MSO

Given two equally long strings $s = \alpha_1 \cdots \alpha_n$ and $s' = \alpha'_1 \cdots \alpha'_n$ of sets α_i and α'_i, let use define the *superposition* $s\&s'$ *of* s *and* s' to be the string obtained by their componentwise unions $\alpha_i \cup \alpha'_i$

$$\alpha_1 \cdots \alpha_n \ \& \ \alpha'_1 \cdots \alpha'_n \ := \ (\alpha_1 \cup \alpha'_1) \ \cdots \ (\alpha_n \cup \alpha'_n).$$

For example,

$$\boxed{\text{E}}\boxed{\text{E}}\boxed{\text{E}} \;\&\; \boxed{}\boxed{\text{R}}\boxed{} \;=\; \boxed{\text{E}}\boxed{\text{E,R}}\boxed{\text{E}}.$$

Extending the operation to sets L and L' of strings of sets, the *superposition $L\&L'$ of L and L'* is the set of superpositions of strings of the same length from L and L'

$$L \;\&\; L' \;:=\; \{s\&s' \mid (s,s') \in L \times L' \text{ and length}(s) = \text{length}(s')\}$$

allowing us to conflate a string s with its singleton language $\{s\}$ (making $s\&s' = \emptyset$ in case s and s' differ in length). Given finite automata accepting L and L', the usual product construction on finite automata for their intersection $L \cap L'$ (e.g. Hopcroft & Ullman 1979) can be adjusted to combine transitions $\rightarrow_L$ for L and $\rightarrow_{L'}$ for L' to form non-deterministic transitions

$$(q,q') \overset{\alpha\cup\alpha'}{\rightarrow} (r,r') \quad\Longleftrightarrow\quad q \overset{\alpha}{\rightarrow}_L r \;\text{ and }\; q' \overset{\alpha'}{\rightarrow}_{L'} r'$$

for $L\&L'$ in lockstep but with labels that may differ. We will loosen the lockstep requirement in §3.2, but first consider constraints that we might impose on superposition (analogous to C on unify$_C(G,G')$ in section 2).

For example, we may wish to require that a fluent a is never followed by a fluent b, as expressed by the predicate logic formula

$$(\forall x)(\forall y)(P_a(x) \wedge S(x,y) \supset \neg P_b(y))$$

where x and y range over string positions, and

$$S(x,y) \text{ says: next after position } x \text{ is } y$$

while for every fluent $a \in \Theta$,

$$P_a(x) \text{ says: } a \text{ occurs at position } x.$$

More precisely, given a string $s = \alpha_1 \cdots \alpha_n$ of n sets α_i of fluents, we can interpret S as the binary relation

$$S_n \;:=\; \{(1,2),(2,3),\ldots,(n-1,n)\}$$

on the set

$$[n] \;:=\; \{1,2,\ldots,n\}$$

of integers from 1 to n, and P_a as the subset

$$P_a^s \;:=\; \{i \in [n] \mid a \in \alpha_i\} \qquad (\text{where } s = \alpha_1 \cdots \alpha_n)$$

of $[n]$, for each fluent a. That is, a string $s \in (2^\Theta)^n$ specifies a structure

$$M_s \;:=\; \langle [n], S_n, \{P_a^s\}_{a\in\Theta} \rangle$$

against which to interpret predicate logic formulas built from S and the P_a's such as the *formulas φ of Monadic Second-Order Logic* (MSO;

e.g. Libkin 2010) generated by the seven clauses

$$\varphi \quad ::= \quad S(x,y) \;\mid\; P_a(x) \;\mid\; X(x) \;\mid\; \varphi \vee \varphi' \;\mid\; \neg\varphi \;\mid\; \exists x\varphi \;\mid\; \exists X\varphi$$

from three disjoint infinite sets Var_1, Var_2 and Θ of first-order variables $x, y \in Var_1$, second-order variables $X \in Var_2$, and fluents $a \in \Theta$, respectively. For any such MSO-formula φ, only finitely many fluents may occur in φ, which we collect in φ's *vocabulary*, $voc(\varphi) \in Fin(\Theta)$

$$\begin{aligned}
voc(S(x,y)) &= voc(X(x)) \;=\; \emptyset \\
voc(P_a(x)) &= \{a\} \\
voc(\varphi \vee \varphi') &= voc(\varphi) \cup voc(\varphi') \\
voc(\neg\varphi) &= voc(\exists x\varphi) \;=\; voc(\exists X\varphi) \;=\; voc(\varphi).
\end{aligned}$$

An MSO-*sentence* is understood to be an MSO-formula in which all variable occurrences are bound. For every $A \in Fin(\Theta)$, we put every MSO sentence with vocabulary contained in A into the set $MSO(A)$

$$MSO(A) := \{\varphi \mid \varphi \text{ is an MSO-sentence and } voc(\varphi) \subseteq A\}$$

and define a binary relation

$$\models_A \;\subseteq\; (2^A)^* \times MSO(A)$$

between $(2^A)^*$ and $MSO(A)$ in the usual Tarskian manner, associating a string $s \in (2^A)^*$ with M_s. (Apologies for reusing the symbol $\models$.) For any string s of sets of fluents, let the *A-reduct* $\rho_A(s)$ *of* s be the componentwise intersection of s with A

$$\rho_A(\alpha_1 \cdots \alpha_n) \;:=\; (\alpha_1 \cap A) \cdots (\alpha_n \cap A)$$

(so-called because $\rho_A(s)$ is precisely the part of s needed to extract from M_s its A-reduct $\langle [n], S_n, \{P^s_a\}_{a \in A}\rangle$).

Fact 2 *For all $A \in Fin(\Theta)$, $\varphi \in MSO(A)$ and $s \in (2^A)^*$,*

$$s \models_A \varphi \quad \Longleftrightarrow \quad \rho_{voc(\varphi)}(s) \models_{voc(\varphi)} \varphi \; .$$

With Fact 2, the relations $\{\models_A\}_{A \in Fin(\Theta)}$ become an institution with signature category $Fin(\Theta)$ provided we

(i) extend the map $A \mapsto MSO(A)$ to pairs (A, A') such that $A \subseteq A' \in Fin(\Theta)$, setting $MSO(A, A')$ to the inclusion $MSO(A) \hookrightarrow MSO(A')$ mapping $\varphi \in MSO(A) \subseteq MSO(A')$ to itself, and

(ii) turn the map $A \mapsto (2^A)^*$ into a contravariant functor $\mathbf{M}$ from $Fin(\Theta)$ so that whenever $A \subseteq A' \in Fin(\Theta)$, $\mathbf{M}(A', A) : (2^{A'})^* \to (2^A)^*$ is the restriction of ρ_A to $(2^{A'})^*$

$$\mathbf{M}(A', A)(s) \;=\; \rho_A(s) \qquad \text{for all } s \in (2^{A'})^*.$$

Büchi's theorem equating sentences in$\mathit{MSO}(A)$ with regular languages over A (e.g. Libkin 2010, page 124) holds also in the present set-up for languages over 2^A (the advantage of 2^A over A being the availability of reducts for Fact 2).

23.3.2 Compression, Branching and Superposition Modified

A string $s \in (2^A)^*$ is understood above to have granularity A. Variations in A are described in Fact 2 that preserve string length using A-reducts. It is natural, however, to expect the length of a string to grow with A, as hinted by the discussion above of

$$s_A \ := \ \boxed{\text{Jan}}\boxed{\text{Feb}}\cdots\boxed{\text{Dec}}$$

and

$$s_{A'} \ := \ \boxed{\text{Jan,d1}}\boxed{\text{Jan,d2}}\cdots\boxed{\text{Jan,d31}}\boxed{\text{Feb,d1}}\cdots\boxed{\text{Dec,d31}}.$$

Put the other way around, the A-reduct of $s_{A'}$

$$\rho_A(s_{A'}) \ = \ \boxed{\text{Jan}}^{31}\boxed{\text{Feb}}^{29}\cdots\boxed{\text{Dec}}^{d31}$$

has substrings such as $\boxed{\text{Jan}}^{31}$ which we might compress to $\boxed{\text{Jan}}$ for

$$\mathit{bc}(\rho_A(s_{A'})) \ = \ \boxed{\text{Jan}}\boxed{\text{Feb}}\cdots\boxed{\text{Dec}} \ = \ s_A$$

where for any string s, $\mathit{bc}(s)$ compresses blocks α^n of $n > 1$ consecutive occurrences in s of the same symbol α to a single α, leaving s otherwise unchanged

$$\mathit{bc}(s) \ := \ \begin{cases} \mathit{bc}(\alpha s') & \text{if } s = \alpha\alpha s' \\ \alpha \ \mathit{bc}(\alpha' s') & \text{if } s = \alpha\alpha' s' \text{ with } \alpha \neq \alpha' \\ s & \text{otherwise.} \end{cases}$$

To require that time progress only with change (discernible at some bounded granularity A), let us work with strings $\alpha_1\alpha_2\cdots\alpha_n$ that are *stutter-free* in that $\alpha_i \neq \alpha_{i+1}$ for i from 1 to $n-1$. That is,

$$\text{a string } s \text{ is stutter-free} \quad \Longleftrightarrow \quad s = \mathit{bc}(s).$$

The restriction of bc to any finite alphabet is computable by a finite-state transducer, as are, for all $A' \in \mathit{Fin}(\Theta)$ and $A \subseteq A'$, the composition $\rho_A; \mathit{bc}$ for bc_A

$$\mathit{bc}_A(s) \ := \ \mathit{bc}(\rho_A(s)) \qquad \text{for } s \in (2^{A'})^*.$$

Without the compression bc in bc_A, we are left with the map ρ_A that leaves the ontology intact (insofar as the domain of an MSO-model is given by the string length), whilst restricting the vocabulary (for A-reducts). The institution described by Fact 2 can be adjusted to another institution in which

- the models are stutter-free strings[3]
- the reducts ρ_A are replaced by bc_A, and
- the satisfaction relations $\models'_A$ are given by explicitly referring to the sentence's vocabulary

$$s \models'_A \varphi \iff bc_{voc(\varphi)}(s) \models_{voc(\varphi)} \varphi.$$

Compressing strings via bc_A allows us to lengthen the strings by inversion. The *inverse limit* $\mathrm{IL}(\Theta, bc)$ *of* Θ, bc consists of functions $\mathbf{a} : \mathit{Fin}(\Theta) \to \mathit{Fin}(\Theta)^*$ that respect the projections bc_A

$$\mathbf{a}(A) = bc_A(\mathbf{a}(A')) \quad \text{whenever} \quad A \subseteq A' \in \mathit{Fin}(\Theta).$$

The prefix relation on strings

$$s \text{ prefix } s' \iff s' = s\hat{s} \text{ for some } \hat{s}$$

lifts to maps $\mathbf{a}$ and $\mathbf{a}'$ in $\mathrm{IL}(\Theta, bc)$ by universal quantification for an irreflexive relation

$$\mathbf{a} \prec \mathbf{a}' \iff \mathbf{a} \neq \mathbf{a}' \text{ and } (\forall A \in \mathit{Fin}(\Theta))\ \mathbf{a}(A) \text{ prefix } \mathbf{a}'(A)$$

that is *tree-like* on $\mathrm{IL}(\Theta, bc)$ — i.e., transitive and left linear: for every $\mathbf{a} \in \mathrm{IL}(\Theta, bc)$, and all $\mathbf{a}_1 \prec \mathbf{a}$ and $\mathbf{a}_2 \prec \mathbf{a}$,

$$\mathbf{a}_1 \prec \mathbf{a}_2 \quad \text{or} \quad \mathbf{a}_2 \prec \mathbf{a}_1 \quad \text{or} \quad \mathbf{a}_2 = \mathbf{a}_1.$$

In other words, time branches at the inverse limit $\mathrm{IL}(\Theta, bc)$.

Even if the strings we are interested in are stutter-free, strings that are not stutter-free can be useful. For instance, to relax the requirement of $L\&L'$ that L and L' run in lockstep, let us collect the strings bc-equivalent to a string in L in

$$L^{bc} \ := \ \{s \in (2^{\Theta})^* \mid (\exists s' \in L)\ bc(s) = bc(s')\}$$

and define the *bc-superposition* $L\&_{bc}L'$ *of* L *and* L' to be the image under bc of the superposition of L^{bc} and ${L'}^{bc}$

$$L \ \&_{bc} \ L' \ := \ \{bc(s) \mid s \in L^{bc} \ \& \ {L'}^{bc}\}$$

(a regular language, if L and L' are). Then for any two fluents $a, a' \in \Theta$, the bc-superposition $\boxed{\,}\boxed{a}\boxed{\,} \ \&_{bc} \ \boxed{\,}\boxed{a'}\boxed{\,}$ is the set

$$\{bc(s) \mid s \in (2^{\{a,a'\}})^* \text{ and } bc_{\{a\}}(s) = \boxed{\,}\boxed{a}\boxed{\,} \text{ and } bc_{\{a'\}}(s) = \boxed{\,}\boxed{a'}\boxed{\,}\}$$

consisting of 13 strings, one for each interval relation in Allen 1983. More generally, for any finite set $A = \{a_1, \ldots, a_n\} \in \mathit{Fin}(\Theta)$ of fluents,

[3]Apart from applying bc, a string can also be made stutter-free by superposition with $(\boxed{\,}\boxed{\text{tic}})^*(\boxed{\,} + \epsilon)$ for some fresh fluent tic. The crucial point is that stutter-freeness ensures the vocabulary is large enough to express the distinctions of interest (lengthening a string if necessary).

the bc-superposition

$$\boxed{}\boxed{a_1}\boxed{} \;\&_{bc}\; \cdots \;\&_{bc}\; \boxed{}\boxed{a_n}\boxed{}$$

represents the event structures over A in the sense of Russell-Wiener (Kamp & Reyle 1993, Fernando 2015).

23.3.3 Taking Stock

What are we to make of the difference between the institutions in sections 2 and 3? At its simplest, the difference is between, on the one hand, a program or automaton (as piece of code) and, on the other hand, an execution or run of it — a modern incarnation of the Aristotelian dichotomy between potentiality and actuality. Focusing on applications to natural language semantics, Table 3 lists contrasts based not only on the widespread encoding of linguistic resources as feature structures (including frames), but also on the notion defended in Carlson 1995 that the truth of a generic statement rests not on "the episodic instances but rather the causal forces behind those instances" (page 225), as well as the distinction between *individual-level* and *stage-level* predicates (Carlson 1977).

section 2	section 3
automata	run
resource	use
generic	episodic
causal	temporal
force	event
universal	particular/instance
individual-level	stage-level

Table 3

Much work remains to flesh out Table 3, and win over the skeptical reader. At stake in Table 3 is justification for viewing the directed graphs in F&V as finite automata.[4]

References

Allen, James F. 1983. Maintaining knowledge about temporal intervals. In *Communications of the ACM*, vol. 26, pages 832–843.

Barwise, Jon. 1974. Axioms for abstract model theory. *Annals of Mathematical Logic* 7:221–265.

[4]My thanks to Cleo Condoravdi for inviting me to contribute to this Festschrift, András Kornai for feedback on this paper, and, not to forget, Lauri Karttunen for setting standards towards which to aspire.

Barwise, Jon and Larry Moss. 1996. *Vicious Circles: On the Mathematics of Non-Wellfounded Phenomena.*. CSLI.

Blackburn, Patrick. 1993. Modal logic and attribute value structures. In M. de Rijke, ed., *Diamonds and Defaults*, pages 19–65. Kluwer.

Brzozowski, Janusz A. 1964. Derivatives of regular expressions. *Journal of the ACM* pages 481–494.

Carlson, Greg N. 1977. A unified analysis of the English bare plural. *Linguistics & Philosophy* 1:413–458.

Carlson, Greg N. 1995. Truth conditions of generic sentences: Two contrasting views. In *The Generic Book*, pages 224–237. University of Chicago Press.

Cooper, Robin. 2012. Type theory and semantics in flux. In R. Kempson, T. Fernando, and N. Asher, eds., *Philosophy of Linguistics*, pages 271–323. North-Holland.

Diaconescu, Răzvan. 2012. Three decades of institution theory. In J.-Y. Beziau, ed., *Universal Logic: An Anthology*, pages 309–322. Springer.

Fernando, Tim. 2015. The semantics of tense and aspect: A finite-state perspective. In S. Lappin and C. Fox, eds., *The Handbook of Contemporary Semantic Theory, Second Edition*, pages 203–236. Wiley.

Fernando, Tim. 2016. Types from frames as finite automata. In A. Foret, G. Morrill, R. Muskens, and R. Osswald, eds., *Formal Grammar 2015/2016*, pages 19–40. Springer.

Fillmore, Charles J. 1982. Frame semantics. In *Linguistics in the Morning Calm*, pages 111–137. Hanshin Publishing Co.

Goguen, Joseph and Rod Burstall. 1992. Institutions: Abstract model theory for specification and programming. *Journal of the ACM* 39:95–146.

Hennessy, Matthew and Robin Milner. 1985. Algebraic laws for non-determinism and concurrency. *Journal of the ACM* 32:137–161.

Hopcroft, John and Jeffrey Ullman. 1979. *Inroduction to Automata Theory, Languages, and Computation*. Addison-Wesley.

Kamp, Hans and Uwe Reyle. 1993. *From Discourse to Logic*. Kluwer.

Karttunen, Lauri. 1984. Features and values (F&V). In *COLING '84*, pages 28–33.

Karttunen, Lauri. 2007. Word play. *Computational Linguistics* 33:443–467.

Kornai, András. 2017. Truth or *dare*. This volume.

Kutz, Oliver, Till Mossakowski, and Dominik Lücke. 2010. Carnap, goguen, and the hyperontologies: Logical pluralism and heterogeneous structuring in ontology design. *Logica Universalis* 4:255–333.

Libkin, Leonid. 2010. *Elements of Finite Model Theory*. Springer.

Shan, Chung-chieh. 2015. Splitting hairs. In *Proceedings of the 20th Amsterdam Colloquium*, pages 363–367.

Tarlecki, Andrzej, Rod Burstall, and Joseph Goguen. 1991. Some fundamental algebraic tools for the semantics of computation: Part 3 indexed categories. *Theoretical Computer Science* pages 239–264.