MSO with tests and reducts

Tim Fernando, David Woods & Carl Vogel (DUBLIN)

Dresden, 25 Sep 2019 (FSMNLP)

- $aab \models \exists x(P_a(x) \wedge \exists y(xSy \wedge P_b(y)))$ % $\Sigma^* ab \Sigma^*$
- $[\![\varphi?]\!] = \{(q,q) \mid q \in [\![\varphi]\!]\}$ Dynamic logic

  $q \in (b_1 + \overline{b_1}) \cdots (b_n + \overline{b_n})$ Kleene algebra with tests (Kozen)

  $$\text{e.g.,} \quad [\![b_1]\!] = b_1(b_2 + \overline{b_2}) \cdots (b_n + \overline{b_n})$$

- for all models M and sentences φ,

  $$M \models \varphi \iff M \restriction \text{vocabulary}(\varphi) \models \varphi$$

Natural language events (Davidson . . .)

Reichenbach

Amundsen flew to the North Pole

fly(A,NP)		*speech*

Amundsen reach the North Pole

fly_0(A,NP)	fly_1(A,NP), reach(A,NP)

Events as partial descriptions of intervals — James Allen

Statives as the basis of Dowty's aspect calculus

events in NL	KAT (Kozen)
statives	tests/Booleans
non-statives	actions

Main points

- MSO_A-model = string $s \in (2^A)^+$ (rather than alphabet A)

$$\rho_B(\alpha_1 \cdots \alpha_n) \ := \ (\alpha_1 \cap B) \cdots (\alpha_n \cap B)$$

- compression bc from test-as-program to test-as-formula

$$s\alpha\alpha s' \ \rightsquigarrow \ s\alpha s'$$

Times$_S$ as change$_B$

$$xSy \ \supset \ \neg \bigwedge_{a \in B} (P_a(x) \equiv P_a(y)) \qquad (\dagger)$$

$$s \models \forall x \forall y (\dagger) \quad \Longleftrightarrow \quad bc(\rho_B(s)) = \rho_B(s)$$

Outline

§1 Transitions and reducts
- guarded strings in MSO

§2 Tests and observable change

§3 Projections and superpositions

Transitions $q_1 \xrightarrow{p_1} q_2 \xrightarrow{p_2} \cdots q_n$

Kleene Algebra:	$p_1 p_2 \cdots p_{n-1} \in \Sigma^+$	(labels)
KA with tests:	add states $q_i \subseteq B$	

	alphabet	$\{b_2\} \xrightarrow{p} \emptyset$
$q_1 p_1 q_2 p_2 \cdots q_n$	$2^B \cup \Sigma$	$\{b_2\}\ p\ \emptyset$
$\hat{q}_1 p_1 \hat{q}_2 p_2 \cdots \hat{q}_n$	$B \cup \overline{B} \cup \Sigma$	$\overline{b_1} b_2\ p\ \overline{b_1}\ \overline{b_2}$
$(q_1 \cup \{p_1\})(q_2 \cup \{p_2\}) \cdots q_n$	$2^{B \cup \Sigma}$	$\boxed{b_2, p}\boxed{}$

$$2^{\{b_1,\ldots,b_k\}} \cong (b_1 + \overline{b_1}) \cdots (b_k + \overline{b_k})$$

Concatenation modified

$$s\hat{q} \diamond_k \hat{q'} s' \simeq s\hat{q}s' \quad \text{if } q = q'$$
$$sq \bullet_\Sigma \alpha s' \simeq s\alpha s' \quad \text{if } q = \alpha \setminus \Sigma$$

Reducts and guarded strings

$$\rho_C(\alpha_1 \cdots \alpha_n) := (\alpha_1 \cap C) \cdots (\alpha_n \cap C)$$

$$\Sigma_\square := \{\boxed{p} \mid p \in \Sigma\}$$
$$\mathcal{G}^B_\Sigma := \{s \in (2^{B \cup \Sigma})^+ \mid \rho_\Sigma(s) \in {\Sigma_\square}^* \square\}$$
$$= \{s \in (2^{B \cup \Sigma})^+ \mid s \models \forall x\ \chi_\Sigma(x)\}$$

$\chi_\Sigma(x)$ says

(i) x is non-final iff some $a \in \Sigma$ occurs in x

$$\exists y(xSy) \equiv \bigvee_{a \in \Sigma} P_a(x)$$

(ii) *no* two symbols from Σ occur in x

$$\neg \bigvee_{a \in \Sigma} (P_a(x) \wedge \bigvee_{a' \in \Sigma \setminus \{a\}} P_{a'}(x))$$

Outline

Tests as programs executing in isolation

$$[\![b]\!] \;:=\; \{q \mid q \subseteq B \text{ and } b \in q\}$$

$$[\![b?]\!] \;:=\; \{(q \cup \{b?\})q \mid q \in [\![b]\!]\}$$

$$P_{b?}(x) \wedge xSy \;\;\supset\;\; P_b(x) \;\wedge\; x \equiv_B y$$

$$\begin{aligned} x \equiv_B y \;\; &:= \;\; \bigwedge_{a \in B} (P_a(x) \equiv P_a(y)) \\ &\equiv \; \neg \bigvee_{a \in B} (P_a(x) \wedge \neg P_a(y)) \vee (P_a(y) \wedge \neg P_a(x)) \end{aligned}$$

$$(q \cup \{b?\})q \;\overset{\rho_B}{\rightsquigarrow}\; qq \;\overset{bc}{\rightsquigarrow}\; q$$

Compressing reducts

s is *stutterless* if $s = bc(s)$ where

$$bc(s) \ := \ s \ \text{ if length}(s) < 2$$

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

$$s \models \forall x \forall y \ (xSy \supset x \not\equiv_B y) \quad \Longleftrightarrow \quad \rho_B(s) \text{ is stutterless}$$

$$s \models \langle C \rangle \varphi \quad \Longleftrightarrow \quad bc(\rho_C(s)) \models \varphi$$

satisfaction condition for *institution* (Goguen & Burstall)

Outline

§1 Transitions and reducts
- guarded strings in MSO

§2 Tests and observable change
- compressing reducts

§3 Projections and superpositions
- partial overlapping descriptions

Superposing projections

For $C \subseteq A$,

$$\mathcal{L}_C := \{bc(s) \mid s \in (2^C)^+\}.$$

For $s \in \mathcal{L}_C$ and $s' \in \mathcal{L}_{C'}$,

$$s \,\&_{C,C'}\, s' \;:=\; \{s'' \in \mathcal{L}_{C \cup C'} \mid bc(\rho_C(s'')) = s \text{ and } bc(\rho_{C'}(s'')) = s'\}$$

$$\fbox{}\fbox{b}\fbox{} \;\&_{\{b\},\{b'\}}\; \fbox{}\fbox{b'}\fbox{} \;=\; \fbox{}\fbox{b, b'}\fbox{} + \fbox{}\fbox{b}\fbox{b'}\fbox{} + \cdots$$

13 Allen interval relations

$$\fbox{}\fbox{b}\fbox{b'}\fbox{} \;\&_{\{b,b'\},\{b',b''\}}\; \fbox{}\fbox{b', b''}\fbox{b'}\fbox{} \;=\; \fbox{}\fbox{b}\fbox{b', b''}\fbox{b'}\fbox{}$$

transitivity table (Allen)

$\&_{C,C'}$ inductively

$$\frac{}{\&(\epsilon, \epsilon, \epsilon)} \text{ (s0)}$$

$$\frac{\&(s, s', s'')}{\&(\alpha s, \alpha' s, (\alpha \cup \alpha')s'')} \text{ (s1)}$$

$$\frac{\&(\alpha s, s', s'')}{\&(\alpha s, \alpha' s', (\alpha \cup \alpha')s'')} \text{ (b1)}$$

$$\frac{\&(s, \alpha' s', s'')}{\&(\alpha s, \alpha' s', (\alpha \cup \alpha')s'')} \text{ (b2)}$$

For $C \cap C' \neq \emptyset$, impose side conditions

$$\alpha \cap C' \subseteq \alpha' \quad \text{and} \quad \alpha' \cap C \subseteq \alpha$$

on (s1), (b1), (b2)

So what?

$s \mathbin{\&}_{C,C'} s'$ combines partial overlapping descriptions $(C, s), (C', s')$

MSO$_A$-model $\hat{s} \in (2^A)^+$ is C-approximated by s and
C'-approximated by s' (for $C, C' \subseteq B$)

$$\hat{s} \text{ is } C\text{-approximated by } s \iff \mathit{bc}(\rho_C(\hat{s})) = s$$

$$\rho_B(\hat{s}) = \mathit{bc}(\rho_B(\hat{s})) \iff \hat{s} \models \forall x \forall y (xSy \supset x \not\equiv_B y)$$

$$x \not\equiv_B y \;\equiv\; \bigvee_{a \in B} \underbrace{(P_a(x) \wedge \neg P_a(y))}_{P_{r(a)}(x)} \vee \underbrace{(P_a(y) \wedge \neg P_a(x))}_{P_{l(a)}(x)}$$

actions localised to a

T h a n k   Y o u

☺