Link Semantic Models

Objective

To deal with various programming features in a uniform way

- How to select proper observation type.
- How to link various observation spaces
- How to derive semantic functions
- How to ensure the model extension preserves algebraic laws

Assignment

1. $sem1(x := e) \;=\; x' = e$

2. $sem2(x := e) \;=\; \mathcal{D}(e) \vdash (x' = e)$

   where $\mathcal{D}(e)$ is true when evaluation of e terminates.

3. $sem3(x := e) \;=$

$$II \lhd eflag \rhd \left(\begin{array}{c} (true \vdash x' = e \wedge \neg eflag') \\ \lhd \mathcal{D}(e) \rhd \\ (true \vdash x' = x \wedge eflag') \end{array} \right)$$

   where $eflag$ is an error indicator

Choice

1. $sem1(x := e \sqcap x := f) \; = \; (x' = e) \vee (x' = f)$

2. $sem2(x := e \sqcap x := f) \; =$

$$(\mathcal{D}(e) \wedge \mathcal{D}(f)) \vdash (x' = e \vee x' = f)$$

3. $sem3(x := e \sqcap x := f) =$

$$II \lhd eflag \rhd true \vdash \left(\begin{array}{c} \mathcal{D}(e) \wedge (x' = e) \wedge \neg eflag' \vee \\ \\ \mathcal{D}(f) \wedge (x' = f) \wedge \neg eflag' \vee \\ \\ \neg(\mathcal{D}(e) \vee \mathcal{D}(f)) \wedge \\ \\ (x' = x) \wedge eflag' \end{array} \right)$$

How to understand complicated languages

- Start with the core of a language
- Add to it, one at a time, a number of new features that are required.
- Ideally, the properties of programs established in the simpler theories of programming can remain valid in the enriched ones.

Toward a uniform treatment

- Select an observation space for the new feature
- Characterise the defining properties of new feature
- Link observation spaces by an enbedding mapping
- Derive the new semantic function by examining the corresponding commuting equation

Enriched Type

Let **D1** be the family of relations over the base type S, and $sem1$ a semantic function of the language **L** over the domain **D1**. To introduce a new feature to **L** we construct an enriched type

$$T \ =_{df} \ extend(S)$$

and its link with the original type S

$$\rho : S \leftrightarrow T$$

Commuting Diagram

The second step is to characterise the defining properties of programs in the enriched domain **D2**. Such algebraic laws, often called *healthiness conditions*, are only valid for real programs.

Let $\mathbf{D2} =_{df} T \leftrightarrow T$. The semantic function $sem2 : \mathbf{L} \rightarrow \mathbf{D2}$ is required to establish the commuting diagram

$$\rho; sem2(P) \;=\; sem1(P); \rho$$

$sem2(P)$ is selected among the healthy solutions of the equation

Link

Let $sem1$ and $sem2$ be semantic functions of the programming language. A link $*$ is required to be

1. a monotonic mapping

$$sem2(P) \;=\; sem1(P)^*$$

2. a homomorphism,

$$(sem1(P)\,\underline{op}\,sem1(Q))^* \;=\; sem2(P)\,\underline{op}\,sem2(Q)$$

Existence of Solutions

Under which condition on the relation ρ, the linear equation

$$\rho\,;\,X \;=\; (sem1(P);\rho)$$

Theorem

$\rho; X \;=\; R$ has solutions if and only if $\rho;(\rho\backslash R) \;=\; R$

where $\rho\backslash R$ denotes the **weakest postspecification** of ρ with respect to R:

$$(\rho; X) \;\Rightarrow\; R \;\text{ if and only if }\; X \;\Rightarrow\; \rho\backslash R$$

$\boxed{\textbf{Proof}}$

Proof $\rho; X \;=\; R$

implies $X \Rightarrow \rho\backslash R$

implies $(\rho; X) \Rightarrow (\rho; (\rho\backslash R))$

implies $R \Rightarrow (\rho; (\rho\backslash R))$

implies $\rho; (\rho\backslash R) \;=\; R$

$\rho;(\rho\backslash R) \;=\; R$?

Theorem If there exists Q such that $\rho;Q;R \;=\; R$, then $\rho;(\rho\backslash R) \;=\; R$

Proof

$$
\begin{array}{rl}
 & (\rho;Q;R) \;=\; R \\
\textbf{implies} & (Q;R) \Rightarrow (\rho\backslash R) \\
\textbf{implies} & \rho;(Q;R) \;\Rightarrow\; \rho;(\rho\backslash R) \\
\textbf{implies} & R \;\Rightarrow\; \rho;(\rho\backslash R)
\end{array}
$$

Corollary $\forall R \bullet\; \rho;(\rho\backslash R) \;=\; R$ if and only if $\rho;(\rho\backslash id) \;=\; id$

How to Calculate $sem2(P)$

Theorem

If $\rho;(\rho\backslash id) \;=\; id$ then

$$sem2(P) \;=\; \rho\backslash(sem1(P);\rho) \;=$$

$$\neg(\breve{\rho};true) \;\vee\; (\breve{\rho};sem1(P);\rho)$$

where $\breve{\rho}$ denotes the converse of the relation ρ

Distributivity

Theorem

If $\rho;(\rho\backslash id_S) \;=\; id_S$, then

(1) $\rho\backslash(R_1 \vee R_2) \;=\; (\rho\backslash R_1) \;\vee\; (\rho\backslash R_2))$

(2) $\rho\backslash((R_1; R_2); \rho) \;=\; (\rho\backslash(R1; \rho)); (\rho\backslash(R2; \rho))$

$\boxed{\textbf{Equation } (X;P) \ = \ R}$

Theorem

$X;P \ = \ R$ has solutions if and only if $\breve{P};X \ = \breve{R}$ does so.

Theorem

The equation $X;P \ = \ R$ has solutions if and only if

$(R/P);P \ = \ R$

where R/P denotes the weakest prespecification of P with respect to R.

Proof

Proof

$$
\begin{aligned}
& X\,;P \;=\; R \text{ has solutions} \\
\equiv\;& \breve{P};X \;=\; \breve{R} \text{ has solutions} \\
\equiv\;& \breve{P};(\breve{P}\backslash\breve{R}) \;=\; \breve{R} \\
\equiv\;& \breve{P};\neg(P;\neg\breve{R}) \;=\; \breve{R} \\
\equiv\;& \breve{P};\neg(\neg R;\breve{P})^{\smile} \;=\; \breve{R} \\
\equiv\;& (R/P);P \;=\; R
\end{aligned}
$$

Solutions of $(X;P) = R$

Theorem

If $(id/P);P \;=\; id$ then

$$R/P \;=\; \neg(true;\breve{P}) \;\vee\; R;\breve{P}$$

Theorem (Distributivity)

If $(id/P);P \;=\; id$, then

(1) $(R1 \lhd b(s) \rhd R2)/P \;=\; (R1/P) \lhd b(s) \rhd (R2/P)$

(2) $(R_1 \vee R_2)/P \;=\; (R_1/P) \vee (R_2/P)$

Observation: State of Variables

Let $S \ =_{df}\ (VAR \rightarrow VAL)$ be the base type. A program can be modelled by a predicate which represents a binary relation on S.

Program	Meaning
$x := e$	$x' = e \ \wedge \ y' = y \ \wedge ... \wedge z' = z$
skip	$x' = x \ \wedge \ y' = y \ \wedge ... \wedge z' = z$
$P \ \sqcap \ Q$	$P \ \vee \ Q$
$P \lhd b(x) \rhd Q$	$P \wedge b(x) \ \vee \ \neg b(x) \wedge Q$
$P; Q$	$\exists m \bullet (P[m/s'] \wedge Q[m/s])$

Termination

To specify non-termination behaviour we introduce a pair of Boolean variables to denote the relevant observation:

1. ok records the observation that the program has been started.

2. ok' records the observation that the program has terminated. When the program fails to terminate, the value of ok' is not determinable.

Healthiness Conditions

1. $P \;=\; (ok \Rightarrow P)$
2. $[P[false/ok'] \Rightarrow P[true/ok']]$
3. $P \;=\; P; (ok \Rightarrow ok' \wedge x' = x \wedge ..z' = z)$

Theorem

P is heathy iff it has the form $Q \vdash R$

where $Q \vdash R \;=_{df}\; (ok \wedge Q) \Rightarrow (ok' \wedge R)$

Embedding ρ

We extend the base type S by adding the logical variable ok

$$T1 \ =_{df} \ S \times (\{ok\} \rightarrow Bool)$$

Define the embedding ρ from S to $T1$ by

$$\rho \ =_{df} \ ok' \ \wedge \ (x' = x) \ \wedge \ ... \ \wedge \ (z' = z)$$

Enriched Observation

For any predicate P representing a binary relation on S, we define its image P^* on the enriched domain $T \leftrightarrow T$ as the weakest healthy solution of the equation

$$\rho; X \;=\; P; \rho$$

Theorem

$\rho; X \;=\; P; \rho$ has the weakest solution

$$P^* \;=\; true \vdash P$$

New Semantic Function

The new definition of primitive commands of our programming language is given by the following table.

Program	Meaning
skip	$true \vdash (x' = x \ \wedge \ \ldots \ \wedge \ z' = z)$
$x := e$	$true \vdash (x' = e \ \wedge \ \ldots \ \wedge \ z' = z)$

$*$ is a homomorphism

Theorem

(1) $(P;Q)^* \;=\; P^*;Q^*$

(2) $(P \vee Q)^* \;=\; P^* \vee Q^*$

(3) $(P \wedge Q)^* \;=\; P^* \wedge Q^*$

(4) $(P \lhd b \rhd Q)^* \;=\; P^* \lhd b \rhd Q^*$

Reactive Paradigm

For reactive programming paradigms we are required to distinguish a complete terminated computation from an incomplete one that is suspended. The former is used to specify the case where the program has finished its execution, but the latter suggests that the program cannot proceed further without an interaction with its environment. For example, a synchronisation command

$$\mathbf{wait}(v = 0)$$

can not be executed unless the value of v is set to zero, perhaps by some other programs in its environment.

Extended Type and Healthiness Condition

We introduce a Boolean variable $wait$ into the type $T1$

$$T2 \;=_{df}\; T1 \times (\{wait\} \rightarrow Bool)$$

The variable $wait$ takes the value false if and only when the program has completed its execution.

If a program Q is asked to start in a waiting state of its predecessor, it leaves the state unchanged.

$$Q \;=\; II \lhd wait \rhd Q$$

where

$$II \;=_{df}\; true \vdash (wait' = wait \wedge x' = x \wedge .. \wedge \; z' = z)$$

Embedding Mapping

Define $\rho \; =_{df} \; true \vdash (\neg wait' \wedge x' = x \; \wedge .. \wedge \; z' = z)$
For any design d in the domain $T1 \leftrightarrow T1$, we define its image d^* in the extended domain $T2 \leftrightarrow T2$ as the weakest healthy solution of the equation

$$\rho; X \;\; = \;\; d; \rho$$

Theorem

(1) $(b \vdash R)^* \; = \; II \lhd wait \rhd (b \vdash (R \wedge \neg wait'))$

(2) $*$ is a homomorphism

Communication

1. Extended type: $T3 \ =_{df} \ T1 \times (\{tr\} \rightarrow \mathbf{seq}(\mathcal{A}))$

2. Healthiness conditions:

   (1) $P \ = \ P \ \wedge \ (tr \leq tr')$

   (2) $P(tr, tr') \ = \ P(\epsilon, (tr' - tr))$

3. Embedding:

   - $\rho \ =_{df} \ true \vdash (tr' = \epsilon \ \wedge \ x' = x \ \wedge \ ... \ \wedge \ z' = z)$
   - $(b \vdash R)^* \ = \ (b \vdash (R \wedge tr = tr')) \ \wedge \ (tr \ \leq \ tr')$

4. $*$ is a homomorphism

Exception

1. Extended type: $T4 \ =_{df} \ T1 \times (\{eflag\} \rightarrow \mathbf{Bool})$
2. Healthiness conditions:

   $P \ = \ II \lhd eflag \rhd P$
3. Embedding:
   - $\rho \ =_{df} \ true \vdash (\neg efalg' \wedge \ x' = x \ \wedge \ ... \ \wedge \ z' = z)$
   - $(b \vdash R)^* \ = \ II \lhd eflag \rhd (b \vdash (R \wedge \neg eflag'))$
4. $*$ is a homomorphism

Probability

1. Extended Type

$$T5 \ =_{df} \ \{prob : S \to [0,\, 1] \mid \Sigma_{s \in S}\, prob(s) = 1\}$$

2. Embedding

- $\rho(ok,\, prob,\, ok',\, s') \ =_{df} \ true \vdash prob(s') > 0$
- $(b \vdash R)^* \ = \ b \vdash (prob'(R) = 1)$

Primitive Commands

The new definition of primitive commands is given by calculation

Primitive command	Meaning
skip	$true \vdash prob'(s) = 1$
$x := e$	$true \vdash prob'(s[e/x]) = 1$

Distributivity

Theorem

(1) $(d1 \lhd b \rhd d2)^* \; = \; d1^* \lhd b \rhd d2^*$

(2) $(d1 \;\vee\; d2)^* \; = \; d1^* \;\vee\; d2^* \;\vee\; \bigvee_{0<r<1}(d1^* \|_{M_r} d2^*)$

(3) $(d1; d2)^* \; = \; d1^* ; \uparrow d2^*$

where

$(b1 \vdash R1) \|_{M_r} (b2 \vdash R2) \; =_{df}$

$((b1 \wedge b2) \vdash (R1(s,\, prob1') \wedge R1(s,\, prob2')) \; ; \; M_r$

$M_r \; =_{df} \; true \vdash (prob' \, = \, r \times prob1 + (1 - r) \times prob2).$

Conclusion

In fact, there is a retraction to link the enriched model with the old one, because there exists a mapping † such that

$$(P^*)^\dagger = P \qquad \text{and} \qquad (Q^\dagger)^* \sqsupseteq Q$$

- In the state-oriented development methods the embedding ρ is designed to connect abstract data type with its concrete representation.
- In model checking ρ is used for data abstraction by dramatically reducing the size of state space.