Semantyka i weryfikacja programów

Bartosz Klin

(slajdy Andrzeja Tarleckiego)

Instytut Informatyki
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytet Warszawski

http://www.mimuw.edu.pl/~klin pok. 5680
klin@mimuw.edu.pl

Strona tego wykładu:
http://www.mimuw.edu.pl/~klin/sem19-20.html

Program Semantics & Verification

Bartosz Klin

(slides courtesy of Andrzej Tarlecki)

Institute of Informatics
Faculty of Mathematics, Informatics and Mechanics
University of Warsaw

http://www.mimuw.edu.pl/~klin office: 5680
klin@mimuw.edu.pl

This course: http://www.mimuw.edu.pl/~klin/sem19-20.html

Parameters

Parameter passing:

- call by value
- call by variable
- call by name

We will do **static binding only**

$$S \in \mathbf{Stmt} ::= \ldots \mid \mathbf{begin}\ D_V\ D_P\ S\ \mathbf{end}$$
$$\mid \mathbf{call}\ p \mid \mathbf{call}\ p(\mathbf{vr}\ x) \mid \mathbf{call}\ p(\mathbf{vl}\ e) \mid \mathbf{call}\ p(\mathbf{nm}\ e)$$
$$D_V \in \mathbf{VDecl} ::= \mathbf{var}\ x; D_V \mid \varepsilon$$
$$D_P \in \mathbf{PDecl} ::= \mathbf{proc}\ p\ \mathbf{is}\ (S); D_P \mid \mathbf{proc}\ p(\mathbf{vr}\ x)\ \mathbf{is}\ (S); D_P$$
$$\mid \mathbf{proc}\ p(\mathbf{vl}\ x)\ \mathbf{is}\ (S); D_P \mid \mathbf{proc}\ p(\mathbf{nm}\ x)\ \mathbf{is}\ (S); D_P \mid \varepsilon$$

Semantic domains

$$\mathbf{PEnv} = \mathbf{IDE} \to (\mathbf{PROC}_0 + \mathbf{PROC}_1^{\mathsf{vr}} + \mathbf{PROC}_1^{\mathsf{vl}} + \mathbf{PROC}_1^{\mathsf{nm}} + \{??\})$$

$$\mathbf{PROC}_0 = \mathbf{Store} \to (\mathbf{Store} + \{??\})$$

$$\mathbf{PROC}_1^{\mathsf{vr}} = \mathbf{Loc} \to \mathbf{PROC}_0$$

$$\mathbf{PROC}_1^{\mathsf{vl}} = \mathbf{Int} \to \mathbf{PROC}_0$$

$$\mathbf{PROC}_1^{\mathsf{nm}} = (\mathbf{Store} \to (\mathbf{Int} + \{??\})) \to \mathbf{PROC}_0$$

Semantic functions

As before:

$$\mathcal{S} : \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{Store} \to (\mathbf{Store} + \{??\})}_{\mathbf{STMT}}$$

$$\mathcal{D}_P : \mathbf{PDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to (\mathbf{PEnv} + \{??\})}_{\mathbf{PDECL}}$$

Semantic clauses

No parameters

$$\mathcal{S}[\![\textbf{call } p]\!]\ \rho_V\ \rho_P = P \text{ where } P = \rho_P\ p \in \mathbf{PROC}_0$$

$$\mathcal{D}_P[\![\textbf{proc } p \textbf{ is } (S); D_P]\!]\ \rho_V\ \rho_P =$$
$$\mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P = \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P[p \mapsto P]$$

Parameter called by variable

$$\mathcal{S}[\![\textbf{call } p(\textbf{vr } y)]\!]\ \rho_V\ \rho_P = P\ l \text{ where } P = \rho_P\ p \in \mathbf{PROC}_1^{\mathsf{vr}},\ l = \rho_V\ y$$

$$\mathcal{D}_P[\![\textbf{proc } p(\textbf{vr } x) \textbf{ is } (S); D_P]\!]\ \rho_V\ \rho_P =$$
$$\mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P\ l = \mathcal{S}[\![S]\!]\ \rho_V[x \mapsto l]\ \rho_P[p \mapsto P]$$

Parameter called by value

$$\mathcal{S}[\![\textbf{call } p(\textbf{vl } e)]\!]\ \rho_V\ \rho_P\ s = P\ n\ s \text{ where } P = \rho_P\ p \in \mathbf{PROC}_1^{\mathsf{vl}},\ n = \mathcal{E}[\![e]\!]\ \rho_V\ s$$

$$\mathcal{D}_P[\![\textbf{proc } p(\textbf{vl } x) \textbf{ is } (S); D_P]\!]\ \rho_V\ \rho_P =$$

$$\mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where}$$

$$P\ n\ s = \mathcal{S}[\![S]\!]\ \rho'_V\ \rho_P[p \mapsto P]\ s' \text{ where}$$

$$l = s\ next,\ \rho'_V = \rho_V[x \mapsto l],\ s' = s[l \mapsto n, next \mapsto l + 1]$$

Parameter called by name

$$\mathcal{S}[\![\textbf{call } p(\textbf{nm } e)]\!]\ \rho_V\ \rho_P = P\ (\mathcal{E}[\![e]\!]\ \rho_V) \text{ where } P = \rho_P\ p \in \mathbf{PROC}_1^{\mathsf{nm}}$$

$$\mathcal{D}_P[\![\textbf{proc } p(\textbf{nm } x) \textbf{ is } (S); D_P]\!]\ \rho_V\ \rho_P =$$
$$\mathcal{D}_P[\![D_P]\!]\ \rho_V\ \rho_P[p \mapsto P] \text{ where } P\ E = \mathcal{S}[\![S]\!]\ \rho_V[x \mapsto E]\ \rho_P[p \mapsto P]$$

OOOPS!

$$\rho_V[x \mapsto E] \notin \mathbf{VEnv}$$

Corrections necessary!

$$\mathbf{VEnv} = \mathbf{Var} \to (\mathbf{Loc} + (\mathbf{Store} \to (\mathbf{Int} + \{??\})) + \{??\})$$

$$\mathcal{E}[\![x]\!]\ \rho_V\ s = \text{let } v = \rho_V\ x \text{ in if } v \in \mathbf{Loc} \text{ then } s\ v$$
$$\text{if } v \in (\mathbf{Store} \to (\mathbf{Int} + \{??\})) \text{ then } v\ s$$

This allows for evaluation of called-by-name parameters,
but not for assignements to variables passed in such a way

Input/output

TINY^{+++}

$$S \in \textbf{Stmt} ::= \ldots \mid \textbf{read}\ x \mid \textbf{write}\ e$$

Semantic domains

$$\mathbf{Stream} = \mathbf{Int} \times \mathbf{Stream} + \{\mathbf{eof}\}$$
$$\mathbf{Input} = \mathbf{Stream}$$
$$\mathbf{Output} = \mathbf{Stream}$$
$$\mathbf{State} = \mathbf{Store} \times \mathbf{Input} \times \mathbf{Output}$$

Actually:

$$\mathbf{Stream} = (\mathbf{Int} \otimes_L \mathbf{Stream}) \oplus \{\mathbf{eof}\}_\perp$$

where:

$$\mathbf{D} \otimes_L \mathbf{D}' = \mathbf{D} \otimes \mathbf{D}'_\perp$$

Interpretation:

$$\mathbf{Stream} = \mathbf{Int}^* + \mathbf{Int}^\omega$$

Semantic functions

$$\mathcal{E} : \mathbf{Exp} \to \underbrace{\mathbf{VEnv} \to \mathbf{State} \to (\mathbf{Int} + \{??\})}_{\mathbf{EXP}}$$

$$\mathcal{B} : \mathbf{BExp} \to \underbrace{\mathbf{VEnv} \to \mathbf{State} \to (\mathbf{Bool} + \{??\})}_{\mathbf{BEXP}}$$

Only one clause to modify here:

$$\mathcal{E}[\![x]\!]\ \rho_V\ \langle s, i, o\rangle = s\ l \text{ where } l = \rho_V\ x$$

Semantics of statements

$$\mathcal{S} : \mathbf{Stmt} \to \underbrace{\mathbf{VEnv} \to \mathbf{PEnv} \to \mathbf{State} \to (\mathbf{State} + \{??\})}_{\mathbf{STMT}}$$

Again, only one clause to change:

$$\mathcal{S}[\![x := e]\!]\ \rho_V\ \rho_P\ \langle s, i, o\rangle = \langle s[l \mapsto n], i, o\rangle \text{ where } l = \rho_V\ x, n = \mathcal{E}[\![e]\!]\ \rho_V\ \langle s, i, o\rangle$$

(plus a similar change in $\mathcal{D}_V[\![\mathbf{var}\ x; D_V]\!]\ \ldots = \ldots$) and two clauses to add:

$$\mathcal{S}[\![\mathbf{read}\ x]\!]\ \rho_V\ \rho_P\ \langle s, i, o\rangle = \langle s[l \mapsto n], i', o\rangle \text{ where } l = \rho_V\ x, \langle n, i'\rangle = i$$

$$\mathcal{S}[\![\mathbf{write}\ e]\!]\ \rho_V\ \rho_P\ \langle s, i, o\rangle = \langle s, i, \langle n, o\rangle\rangle \text{ where } n = \mathcal{E}[\![e]\!]\ \rho_V\ \langle s, i, o\rangle$$

Cheating a bit: writing out in the reverse order

Programs

New syntactic domain:

$$\mathbf{Prog} ::= \mathbf{prog}\ S$$

with obvious semantic function:

$$\mathcal{P} : \mathbf{Prog} \to \underbrace{\mathbf{Input} \to (\mathbf{Output} + \{??\})}_{\mathbf{PROG}}$$

given by:

$$\mathcal{P}[\![\mathbf{prog}\ S]\!]\ i = o' \text{ where } \mathcal{S}[\![S]\!]\ \rho_V^\emptyset\ \rho_P^\emptyset\ \langle s^\emptyset, i, \mathbf{eof}\rangle = \langle s', i', o'\rangle,$$
$$\rho_V^\emptyset\ x = ??, \rho_P^\emptyset\ p = ??, s^\emptyset\ next = 0, s^\emptyset\ l = ??$$

Okay, but...

Do we want to allow statements to erase output?

Changing philosophy

From: What happens now?

To: What the overall answer will be?

Direct semantics

$$\textbf{begin} \ \ldots \ ; \ \ldots \ ; \ \ldots \ \textbf{end}$$

$$s^{\emptyset} \xrightarrow{\mathcal{S}[\![\ldots]\!]} s_i \xrightarrow{\mathcal{S}[\![\ldots]\!]} s_j \xrightarrow{\mathcal{S}[\![\ldots]\!]} s' \rightsquigarrow \text{“overall result”}$$

Continuation semantics

$$\textbf{begin} \ \ldots \ ; \ \ldots \ ; \ \ldots \ \textbf{end}$$

$$\kappa' : \rightsquigarrow \text{“overall result”}$$

$$\xleftarrow{\mathcal{S}[\![\ldots]\!]} \kappa_i : \rightsquigarrow \text{“overall result”}$$

$$\xleftarrow{\mathcal{S}[\![\ldots]\!]} \kappa_j : \rightsquigarrow \text{“overall result”}$$

$$\xleftarrow{\mathcal{S}[\![\ldots]\!]} \kappa^{\emptyset} : \rightsquigarrow \text{“overall result”}$$