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`

Blocks & declarations

Let's look at declarations of local variables:

$\textsc{Tiny}^{+}$

$$S \in \mathbf{Stmt} ::= \ldots \mid \mathbf{begin}\ D_V\ S\ \mathbf{end}$$

$$D_V \in \mathbf{VDecl} ::= \mathbf{var}\ x; D_V \mid \varepsilon$$

Locations

We have identified two roles of variables:

- identifiers, as used in programs
- names for memory cells, where values are stored

To separate them, the structure of the semantic domains has to be changed.

Splitting states into *environments* and *stores*

$$\begin{aligned}\mathbf{Int} &= \{0, 1, -1, 2, -2, \ldots\}_\perp \\ \mathbf{Bool} &= \{\mathbf{tt}, \mathbf{ff}\}_\perp \\ \mathbf{VEnv} &= \mathbf{Var} \to (\mathbf{Loc} + \{??\}) \\ \mathbf{Store} &= \mathbf{Loc} \to (\mathbf{Int} + \{??\})\end{aligned}$$

$$combine : \mathbf{VEnv} \times \mathbf{Store} \to \mathbf{State}$$

$$combine(\rho_V, s) = \lambda x{:}\mathbf{Var}.s(\rho_V(x))$$

Inaccurate, but right...

Semantic functions

$$\mathcal{N} : \mathbf{Num} \to \mathbf{Int}$$

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

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

and then for instance

$$\mathcal{E}[\![x]\!] = \lambda\rho_V{:}\mathbf{VEnv}.\lambda s{:}\mathbf{Store}.\mathit{ifte}(\rho_V\ x = ??, ??, \mathit{ifte}(s\ (\rho_V\ x) = ??, ??, s\ (\rho_V\ x)))$$

$$\begin{aligned}\mathcal{E}[\![e_1 + e_2]\!] = \lambda\rho_V{:}\mathbf{VEnv}.\lambda s{:}\mathbf{Store}.\mathit{ifte}(&\mathcal{E}[\![e_1]\!]\ \rho_V\ s = ??, ??,\\ &\mathit{ifte}(\mathcal{E}[\![e_2]\!]\ \rho_V\ s = ??, ??,\\ &\quad \mathcal{E}[\![e_1]\!]\ \rho_V\ s + \mathcal{E}[\![e_2]\!]\ \rho_V\ s))\end{aligned}$$

Looks horrible!
Reads even worse!

Bits of notation

- (re)move lambda-abstraction
- use **where**-notation, **let**-notation, explicit **if-then**-notation, etc
- assume that errors ?? propagate

Then:

$$\mathcal{E}[\![x]\!]\ \rho_V\ s = s\ l \text{ where } l = \rho_V\ x$$

$$\mathcal{E}[\![e_1 + e_2]\!]\ \rho_V\ s = n_1 + n_2 \text{ where } n_1 = \mathcal{E}[\![e_1]\!]\ \rho_V\ s, n_2 = \mathcal{E}[\![e_2]\!]\ \rho_V\ s$$

Relate this to the previous semantics using $combine$

Write down all the other rules for $\mathcal{E}$ and $\mathcal{B}$.
Spell out their exact meaning
expanding all the notations in use.

Statements

One could work with "big states"

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

BUT:

Statements do not modify the environment!

Hence:

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

Semantic clauses

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

$$\mathcal{S}[\![\mathbf{skip}]\!]\ \rho_V\ s = s$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V\ s = \mathcal{S}[\![S_2]\!]\ \rho_V\ s_1 \text{ where } s_1 = \mathcal{S}[\![S_1]\!]\ \rho_V\ s$$

$$\begin{aligned}\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]\ \rho_V\ s = {} & \text{let } v = \mathcal{B}[\![b]\!]\ \rho_V\ s \text{ in} \\ & \text{if } v = \mathbf{tt} \text{ then } \mathcal{S}[\![S_1]\!]\ \rho_V\ s \\ & \text{if } v = \mathbf{ff} \text{ then } \mathcal{S}[\![S_2]\!]\ \rho_V\ s\end{aligned}$$

$$\begin{aligned}\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V\ s = {} & \text{let } v = \mathcal{B}[\![b]\!]\ \rho_V\ s \text{ in} \\ & \text{if } v = \mathbf{ff} \text{ then } s \\ & \text{if } v = \mathbf{tt} \text{ then } \mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V\ s' \\ & \qquad \text{where } s' = \mathcal{S}[\![S]\!]\ \rho_V\ s\end{aligned}$$

More compact version

Relying on propagation of errors $??$ to be also built into composition of functions from $\mathbf{Store}$ to $\mathbf{Store} + \{??\}$:

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

$$\mathcal{S}[\![\mathbf{skip}]\!]\ \rho_V = id_{\mathbf{Store}}$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V = \mathcal{S}[\![S_1]\!]\ \rho_V ; \mathcal{S}[\![S_2]\!]\ \rho_V$$

$$\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]\ \rho_V = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S_1]\!]\ \rho_V, \mathcal{S}[\![S_2]\!]\ \rho_V)$$

$$\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S]\!]\ \rho_V ; \mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V, id_{\mathbf{Store}})$$

The missing clause for blocks in a moment

Declarations modify environments

$$\mathcal{D}_V : \mathbf{VDecl} \to \underbrace{\mathbf{VEnv} \to \mathbf{Store} \to (\mathbf{VEnv} \times \mathbf{Store} + \{??\})}_{\mathbf{VDECL}}$$

$$\mathcal{D}_V [\![\varepsilon]\!]\, \rho_V\ s = \langle \rho_V , s \rangle$$

$$\mathcal{D}_V [\![\mathbf{var}\ x; D_V]\!]\, \rho_V\ s = \mathcal{D}_V [\![D_V]\!]\, \rho'_V\ s'$$

where $l = newloc(s), \rho'_V = \rho_V[x \mapsto l], s' = s[l \mapsto ??]$

Trouble: We want $newloc : \mathbf{Store} \to \mathbf{Loc}$ to yield a new, unused location. This cannot be defined under the definitions given so far. Solution: more information in stores is needed to determine used and unused locations.

Simple solution

Take:

$$\mathbf{Loc} = \{0, 1, 2, \ldots\}$$

Add to each store a pointer to the next unused location:

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

Semantic clauses then:

$$\mathcal{D}_V[\![\varepsilon]\!]\,\rho_V\ s = \langle \rho_V, s\rangle$$

$$\mathcal{D}_V[\![\mathbf{var}\ x; D_V]\!]\,\rho_V\ s =$$

$$\mathcal{D}_V[\![D_V]\!]\,\rho'_V\ s' \text{ where } l = s\ next, \rho'_V = \rho_V[x \mapsto l], s' = s[l \mapsto ??, next \mapsto l+1]$$

Semantics of blocks

$$\mathcal{S}[\![\textbf{begin } D_V \ S \textbf{ end}]\!]\ \rho_V\ s = \mathcal{S}[\![S]\!]\ \rho'_V\ s' \text{ where } \langle \rho'_V, s' \rangle = \mathcal{D}_V[\![D_V]\!]\ \rho_V\ s$$

The scope of a declaration is the block it occurs in

with holes resulting from redeclarations of the same variable within it

For instance

$$\textbf{begin var } y; \textbf{var } x\ x := 1; \textbf{begin var } x\ y := 2; x := 5 \textbf{ end}; y := x \textbf{ end}$$

may be marked as follows to indicate the relevant declarations:

$$\textbf{begin } \boxed{\textbf{var } y}; \boxed{\textbf{var } x}\ \boxed{x} := 1; \textbf{begin } \boxed{\textbf{var } x}\ \boxed{y} := 2; \boxed{x} := 5 \textbf{ end}; \boxed{y} := \boxed{x} \textbf{ end}$$

Procedures

$\textsc{Tiny}^{++}$

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

- binding of global variables
- recursion

Binding of global variables

Static binding

$$
\begin{array}{l}
\textbf{begin var } y;\\
\qquad \boxed{\textbf{var } x};\\
\qquad \textbf{proc } p \textbf{ is } (\boxed{x} := 1);\\
\qquad \textbf{begin } \boxed{\textbf{var } x};\\
\qquad\qquad \boxed{x} := 3;\\
\qquad\qquad \textbf{call } p;\\
\qquad\qquad y := \boxed{x}\\
\qquad \textbf{end}\\
\textbf{end}
\end{array}
$$

Dynamic binding

$$
\begin{array}{l}
\textbf{begin var } y;\\
\qquad \boxed{\textbf{var } x};\\
\qquad \textbf{proc } p \textbf{ is } (\boxed{\boxed{x}} := 1);\\
\qquad \textbf{begin } \boxed{\textbf{var } x};\\
\qquad\qquad \boxed{x} := 3;\\
\qquad\qquad \textbf{call } p; \quad \text{\%\%\% with } \boxed{x}\\
\qquad\qquad y := \boxed{x}\\
\qquad \textbf{end}\\
\textbf{end}
\end{array}
$$

Semantic domains and functions

Dynamic binding

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

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

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

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

Semantic clauses

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

$$\mathcal{S}[\![\mathbf{skip}]\!]\ \rho_V\ \rho_P = id_{\mathbf{Store}}$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V\ \rho_P = \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P ; \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P$$

$$\mathcal{S}[\![\mathbf{if}\ b\ \mathbf{then}\ S_1\ \mathbf{else}\ S_2]\!]\ \rho_V\ \rho_P = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P, \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P)$$

$$\mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V\ \rho_P = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P ; \mathcal{S}[\![\mathbf{while}\ b\ \mathbf{do}\ S]\!]\ \rho_V\ \rho_P, id_{\mathbf{Store}})$$

$$\boxed{\mathcal{S}[\![\mathbf{call}\ p]\!]\ \rho_V\ \rho_P = P\ \rho_V\ \rho_P \text{ where } P = \rho_P\ p}$$

$$\mathcal{S}[\![\mathbf{begin}\ D_V\ D_P\ S\ \mathbf{end}]\!]\ \rho_V\ \rho_P\ s = \mathcal{S}[\![S]\!]\ \rho'_V\ \rho'_P\ s' \text{ where } \langle \rho'_V, s' \rangle = \mathcal{D}_V[\![D_V]\!]\ \rho_V\ s,\ \rho'_P = \mathcal{D}_P[\![D_P]\!]\ \rho_P$$

$$\mathcal{D}_P[\![\varepsilon]\!] = id_{\mathbf{PEnv}}$$

$$\boxed{\mathcal{D}_P[\![\mathbf{proc}\ p\ \mathbf{is}\ (S); D_P]\!]\ \rho_P = \mathcal{D}_P[\![D_P]\!]\ \rho_P[p \mapsto \mathcal{S}[\![S]\!]]}$$

Recursion

$$
\begin{array}{l}
\textbf{begin var } x; \\
\qquad \textbf{proc } NO \textbf{ is } (\textbf{if } 101 \leq x \textbf{ then } x := x - 10 \\
\qquad\qquad\qquad\qquad \textbf{else } (x := x + 11; \textbf{call } NO; \textbf{call } NO)\ ); \\
\qquad x := 54; \\
\qquad \textbf{call } NO \\
\textbf{end}
\end{array}
$$

Semantic domains and functions

Static binding

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

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

Semantic clauses

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

$$\mathcal{S}[\![\textbf{skip}]\!]\ \rho_V\ \rho_P = id_{\textbf{Store}}$$

$$\mathcal{S}[\![S_1; S_2]\!]\ \rho_V\ \rho_P = \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P;\mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P$$

$$\mathcal{S}[\![\textbf{if } b \textbf{ then } S_1 \textbf{ else } S_2]\!]\ \rho_V\ \rho_P = cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S_1]\!]\ \rho_V\ \rho_P, \mathcal{S}[\![S_2]\!]\ \rho_V\ \rho_P)$$

$$\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!]\ \rho_V\ \rho_P = \\ cond(\mathcal{B}[\![b]\!]\ \rho_V, \mathcal{S}[\![S]\!]\ \rho_V\ \rho_P;\mathcal{S}[\![\textbf{while } b \textbf{ do } S]\!]\ \rho_V\ \rho_P, id_{\textbf{Store}})$$

$$\boxed{\mathcal{S}[\![\textbf{call } p]\!]\ \rho_V\ \rho_P = P \text{ where } P = \rho_P\ p}$$

$$\mathcal{S}[\![\textbf{begin } D_V\ D_P\ S \textbf{ end}]\!]\ \rho_V\ \rho_P\ s = \\ \mathcal{S}[\![S]\!]\ \rho'_V\ \rho'_P\ s' \text{ where } \langle \rho'_V, s' \rangle = \mathcal{D}_V[\![D_V]\!]\ \rho_V\ s,\ \ \rho'_P = \mathcal{D}_P[\![D_P]\!]\ \rho'_V\ \rho_P$$

$$\mathcal{D}_P[\![\varepsilon]\!]\ \rho_V = id_{\textbf{PEnv}}$$

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