Denotational Semantics Examples

Andrew W. Appel

Straight-line programs

Types:

$$\begin{array}{rcl}
V & = & \mathit{Int} \\
\mathit{Ide} & & \\
S & = & \mathit{Ide} \to \mathit{Int} \\
\mathit{Exp} & = & S \to \mathit{Int} \\
\mathit{Stm} & = & S \to S
\end{array}$$

Functions:

$$\begin{array}{rl}
\text{plus}: & (V \times V) \to V \\
\text{update}: & (S \times \mathit{Ide} \times V) \to S \\
\mathcal{I}: & \texttt{ID} \to \mathit{Ide} \\
\mathcal{N}: & \texttt{NUM} \to V \\
\mathcal{E}: & \texttt{exp} \to \mathit{Exp} \\
\mathcal{S}: & \texttt{stm} \to \mathit{Stm}
\end{array}$$

Grammar rules and semantic equations:

$\texttt{stm} \to \texttt{ID := exp}$	$\mathcal{S}[\texttt{stm}_0] = \lambda s.\text{update}(s, \mathcal{I}[\texttt{ID}], \mathcal{E}[\texttt{exp}_1]s)$
$\texttt{stm} \to \texttt{stm ; stm}$	$\mathcal{S}[\texttt{stm}_0] = \lambda s.\mathcal{S}[\texttt{stm}_2](\mathcal{S}[\texttt{stm}_1]s)$
$\texttt{exp} \to \texttt{NUM}$	$\mathcal{E}[\texttt{exp}_0] = \lambda s.\mathcal{N}[\texttt{NUM}]$
$\texttt{exp} \to \texttt{ID}$	$\mathcal{E}[\texttt{exp}_0] = \lambda s.s(\mathcal{I}[\texttt{ID}])$
$\texttt{exp} \to \texttt{exp + exp}$	$\mathcal{E}[\texttt{exp}_0] = \lambda s.\text{plus}(\mathcal{E}[\texttt{exp}_1]s, \mathcal{E}[\texttt{exp}_2]s)$
$\texttt{exp} \to \texttt{(exp)}$	$\mathcal{E}[\texttt{exp}_0] = \mathcal{E}[\texttt{exp}_1]$

Control Flow

Types:

$$
\begin{aligned}
V &= \mathit{Int} \\
\mathit{Ide} & \\
S &= \mathit{Ide} \to \mathit{Int} \\
\mathit{Exp} &= S \to \mathit{Int} \\
\mathit{Stm} &= S \to S
\end{aligned}
$$

Functions:

$$
\begin{aligned}
\text{plus} : &\quad (V \times V) \to V \\
\text{update} : &\quad (S \times \mathit{Ide} \times V) \to S \\
\text{ifnz} : &\quad (V \times S \times S) \to S \\
\mathcal{I} : &\quad \texttt{ID} \to \mathit{Ide} \\
\mathcal{N} : &\quad \texttt{NUM} \to V \\
\mathcal{E} : &\quad \texttt{exp} \to \mathit{Exp} \\
\mathcal{S} : &\quad \texttt{stm} \to \mathit{Stm}
\end{aligned}
$$

Grammar rules and semantic equations:

`stm` $\to$ `ID := exp`	$\mathcal{S}[\texttt{stm}_0] = \lambda s.\text{update}(s, \mathcal{I}[\texttt{ID}], \mathcal{E}[\texttt{exp}_1]s)$
`stm` $\to$ `stm ; stm`	$\mathcal{S}[\texttt{stm}_0] = \lambda s.\mathcal{S}[\texttt{stm}_2](\mathcal{S}[\texttt{stm}_1]s)$
`stm` $\to$ `IF exp THEN stm ELSE stm`	$\mathcal{S}[\texttt{stm}_0] = \lambda s.\text{ifnz}(\mathcal{E}[\texttt{exp}_1]s, \mathcal{S}[\texttt{stm}_1]s, \mathcal{S}[\texttt{stm}_2]s)$
`stm` $\to$ `WHILE exp DO stm`	$\mathcal{S}[\texttt{stm}_0] = Y(\lambda f.\lambda s.\text{ifnz}(\mathcal{E}[\texttt{exp}_1]s, f(\mathcal{S}[\texttt{stm}_1]s), s))$
`exp` $\to$ `NUM`	$\mathcal{E}[\texttt{exp}_0] = \lambda s.\mathcal{N}[\texttt{NUM}]$
`exp` $\to$ `ID`	$\mathcal{E}[\texttt{exp}_0] = \lambda s.s(\mathcal{I}[\texttt{ID}])$
`exp` $\to$ `exp + exp`	$\mathcal{E}[\texttt{exp}_0] = \lambda s.\text{plus}(\mathcal{E}[\texttt{exp}_1]s, \mathcal{E}[\texttt{exp}_2]s)$
`exp` $\to$ `( exp )`	$\mathcal{E}[\texttt{exp}_0] = \mathcal{E}[\texttt{exp}_1]$

Locations and Environments

Types:

$$
\begin{array}{rcl}
V & = & \mathit{Int} \\
\mathit{Ide} & = & \mathit{Identifiers} \\
L & = & \mathit{Locations} \\
\mathit{Env} & = & \mathit{Ide} \to L \\
S & = & L \to V \\
\mathit{Exp} & = & \mathit{Env} \to S \to V \\
\mathit{Stm} & = & (\mathit{Env} \times L) \to S \to S
\end{array}
$$

Functions:

$$
\begin{array}{rcl}
\text{plus}: & & (V \times V) \to V \\
\text{upds}: & & (S \times L \times V) \to S \\
\text{upde}: & & (\mathit{Env} \times \mathit{Ide} \times L) \to \mathit{Env} \\
l_0: & & L \\
\text{next}: & & L \to L \\
s_0: & & S \\
e_0: & & \mathit{Env} \\
e_1 & = & \texttt{upde}(e_0, \mathcal{I}(\texttt{Answer}), l_0) \\
\mathcal{I}: & & \texttt{ID} \to \mathit{Ide} \\
\mathcal{N}: & & \texttt{NUM} \to V \\
\mathcal{E}: & & \texttt{exp} \to \mathit{Exp} \\
\mathcal{S}: & & \texttt{stm} \to \mathit{Stm} \\
\mathcal{P}: & & \texttt{prog} \to V
\end{array}
$$

Grammar rules and semantic equations:

$$
\begin{array}{ll}
\texttt{prog} \to \texttt{stm} & \mathcal{P}[\texttt{prog}_0] = \mathcal{S}[\texttt{stm}_1](e_1, \text{next}(l_0))\ s_0\ l_0 \\
 & \\
\texttt{stm} \to \texttt{ID := exp} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,l).\lambda s.\text{upds}(s, e(\mathcal{I}[\texttt{ID}]), \mathcal{E}[\texttt{exp}_1] e\ s) \\
\texttt{stm} \to \{\ \texttt{VAR ID ; stm}\ \} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,l).\mathcal{S}[\texttt{stm}_1](\texttt{upde}(e, \mathcal{I}[\texttt{ID}], \texttt{next}(l)), \texttt{next}(l)) \\
\texttt{stm} \to \texttt{stm ; stm} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,l).\lambda s.\mathcal{S}[\texttt{stm}_2](e,l)\ (\mathcal{S}[\texttt{stm}_1](e,l)\ s) \\
 & \\
\texttt{exp} \to \texttt{NUM} & \mathcal{E}[\texttt{exp}_0] = \lambda e.\lambda s.\mathcal{N}[\texttt{NUM}] \\
\texttt{exp} \to \texttt{ID} & \mathcal{E}[\texttt{exp}_0] = \lambda e.\lambda s.s(e(\mathcal{I}[\texttt{ID}])) \\
\texttt{exp} \to \texttt{exp + exp} & \mathcal{E}[\texttt{exp}_0] = \lambda e.\lambda s.\text{plus}(\mathcal{E}[\texttt{exp}_1]\ e\ s, \mathcal{E}[\texttt{exp}_2]\ e\ s) \\
\texttt{exp} \to \texttt{(exp)} & \mathcal{E}[\texttt{exp}_0] = \mathcal{E}[\texttt{exp}_1]
\end{array}
$$

Gotos and Labels

Types:

$$
\begin{array}{rcl}
V & = & \mathit{Int} \\
\mathit{Ide} & = & \mathit{Identifiers} \\
S & = & \mathit{Ide} \to V \\
A & = & \mathit{Answers} \\
C & = & S \to A \\
\mathit{Env} & = & \mathit{Ide} \to C \\
\mathit{Exp} & = & S \to V \\
\mathit{Stm} & = & (\mathit{Env} \times C) \to (\mathit{Env} \times C) \\
\mathit{Prog} & = & A
\end{array}
$$

Functions:

$$
\begin{array}{rl}
\text{plus}: & (V \times V) \to V \\
\text{upds}: & (S \times \mathit{Ide} \times V) \to S \\
\text{upde}: & (\mathit{Env} \times \mathit{Ide} \times C) \to \mathit{Env} \\
\text{ifnz}: & (V \times A \times A) \to S \\
\text{snd}: & (\mathit{Env} \times C) \to C \\
\text{done}: & C \\
\text{s}_0: & S \\
\mathcal{I}: & \texttt{ID} \to \mathit{Ide} \\
\mathcal{N}: & \texttt{NUM} \to V \\
\mathcal{E}: & \texttt{exp} \to \mathit{Exp} \\
\mathcal{S}: & \texttt{stm} \to \mathit{Stm} \\
\mathcal{P}: & \texttt{prog} \to \mathit{Prog}
\end{array}
$$

Grammar rules and semantic equations:

$$
\begin{array}{ll}
\texttt{prog} \to \texttt{stm} & \mathcal{P}[\texttt{prog}_0] = \text{snd}(Y(\lambda(e,c).\mathcal{S}[\texttt{stm}_1](e,\text{done})))\text{s}_0 \\
 & \\
\texttt{stm} \to \texttt{ID := exp} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,c).(e, \lambda s.c(\text{upds}(s, \mathcal{I}[\texttt{ID}], \mathcal{E}[\texttt{exp}_1]s))) \\
\texttt{stm} \to \texttt{ID:} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,c).(\mathit{upde}(e, \mathcal{I}[\texttt{ID}], c), c) \\
\texttt{stm} \to \texttt{stm ; stm} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,c).\mathcal{S}[\texttt{stm}_1](\mathcal{S}[\texttt{stm}_2](e,c)) \\
\texttt{stm} \to \texttt{IF exp GOTO ID} & \mathcal{S}[\texttt{stm}_0] = \lambda(e,c).(e, \lambda s.\text{ifnz}(\mathcal{E}[\texttt{exp}_1]s, e(\mathcal{I}[\texttt{ID}])s, cs)) \\
 & \\
\texttt{exp} \to \texttt{NUM} & \mathcal{E}[\texttt{exp}_0] = \lambda e.\lambda s.\mathcal{N}[\texttt{NUM}] \\
\texttt{exp} \to \texttt{ID} & \mathcal{E}[\texttt{exp}_0] = \lambda e.\lambda s.s(e(\mathcal{I}[\texttt{ID}])) \\
\texttt{exp} \to \texttt{exp + exp} & \mathcal{E}[\texttt{exp}_0] = \lambda e.\lambda s.\text{plus}(\mathcal{E}[\texttt{exp}_1]\ e\ s, \mathcal{E}[\texttt{exp}_2]\ e\ s) \\
\texttt{exp} \to \texttt{(exp)} & \mathcal{E}[\texttt{exp}_0] = \mathcal{E}[\texttt{exp}_1]
\end{array}
$$

Expression Continuations

Types:

$$\begin{aligned}
V &= \mathit{Int} \\
\mathit{Ide} &= \mathit{Identifiers} \\
S &= \mathit{Ide} \to V \\
A &= \mathit{Answers} \\
C &= S \to A \\
K &= V \to C \\
\mathit{Exp} &= K \to C \\
\mathit{Stm} &= C \to C \\
\mathit{Prog} &= A
\end{aligned}$$

Functions:

$$\begin{aligned}
\text{plus}: &\quad (V \times V) \to V \\
\text{upd}: &\quad (S \times \mathit{Ide} \times V) \to S \\
\text{ifnz}: &\quad (V \times A \times A) \to S \\
\text{done}: &\quad V \to A \\
\text{error}: &\quad A \\
\text{s}_0: &\quad S \\
\mathcal{I}: &\quad \texttt{ID} \to \mathit{Ide} \\
\mathcal{N}: &\quad \texttt{NUM} \to V \\
\mathcal{E}: &\quad \texttt{exp} \to \mathit{Exp} \\
\mathcal{S}: &\quad \texttt{stm} \to \mathit{Stm} \\
\mathcal{P}: &\quad \texttt{prog} \to \mathit{Prog}
\end{aligned}$$

Grammar rules and semantic equations:

$\texttt{prog} \to \texttt{stm}$	$\mathcal{P}[\texttt{prog}_0] = \mathcal{S}[\texttt{stm}_1](\lambda s.\text{error})\text{s}_0$
$\texttt{stm} \to \texttt{ID := exp}$	$\mathcal{S}[\texttt{stm}_0] = \lambda c.\mathcal{E}[\texttt{exp}_1](\lambda v.\lambda s.c(\text{upd}(s, \mathcal{I}[\texttt{ID}], v)))$
$\texttt{stm} \to \texttt{RETURN exp}$	$\mathcal{S}[\texttt{stm}_0] = \lambda c.\mathcal{E}[\texttt{exp}_1](\lambda v.\lambda s.\text{done}(v))$
$\texttt{stm} \to \texttt{stm ; stm}$	$\mathcal{S}[\texttt{stm}_0] = \lambda c.\mathcal{S}[\texttt{stm}_1](\mathcal{S}[\texttt{stm}_2]c)$
$\texttt{exp} \to \texttt{(stm, exp)}$	$\mathcal{E}[\texttt{exp}_0] = \lambda k.\mathcal{S}[\texttt{stm}_1](\mathcal{E}[\texttt{exp}_1]k)$
$\texttt{exp} \to \texttt{NUM}$	$\mathcal{E}[\texttt{exp}_0] = \lambda k.k(\mathcal{N}[\texttt{NUM}])$
$\texttt{exp} \to \texttt{ID}$	$\mathcal{E}[\texttt{exp}_0] = \lambda k.\lambda s.k(s(\mathcal{I}[\texttt{ID}]))s$
$\texttt{exp} \to \texttt{exp + exp}$	$\mathcal{E}[\texttt{exp}_0] = \lambda k.\mathcal{E}[\texttt{exp}_1](\lambda v_1.\mathcal{E}[\texttt{exp}_2](\lambda v_2.k(\text{plus}(v_1, v_2))))$
$\texttt{exp} \to \texttt{(exp)}$	$\mathcal{E}[\texttt{exp}_0] = \mathcal{E}[\texttt{exp}_1]$

THROW and CATCH

$$
\begin{array}{rcl}
V & = & Int \\
Ide & = & Identifiers \\
S & = & Ide \to V \\
A & = & Answers \\
C & = & S \to A \\
K & = & V \to C \\
Exp & = & (K \times K) \to C \\
Stm & = & (C \times K) \to C \\
Prog & = & A
\end{array}
$$

Functions:

$$
\begin{array}{rl}
\text{plus}: & (V \times V) \to V \\
\text{upd}: & (S \times Ide \times V) \to S \\
\text{ifnz}: & (V \times A \times A) \to S \\
\text{done}: & V \to A \\
\text{error}: & A \\
\text{s}_0: & S \\
\mathcal{I}: & \texttt{ID} \to Ide \\
\mathcal{N}: & \texttt{NUM} \to V \\
\mathcal{E}: & \texttt{exp} \to Exp \\
\mathcal{S}: & \texttt{stm} \to Stm \\
\mathcal{P}: & \texttt{prog} \to Prog
\end{array}
$$

Grammar rules and semantic equations:

$$
\begin{array}{ll}
\texttt{prog} \to \texttt{stm} & \mathcal{P}[\texttt{prog}_0] = \mathcal{S}[\texttt{stm}_1](\lambda s.\text{error}, \lambda v.\lambda s.\text{error})\text{s}_0 \\
 & \\
\texttt{stm} \to \texttt{ID := exp} & \mathcal{S}[\texttt{stm}_0] = \lambda(c,k).\mathcal{E}[\texttt{exp}_1](\lambda v.\lambda s.c(\text{upd}(s, \mathcal{I}[\texttt{ID}], v)), k) \\
\texttt{stm} \to \texttt{RETURN exp} & \mathcal{S}[\texttt{stm}_0] = \lambda(c,k).\mathcal{E}[\texttt{exp}_1](\lambda v.\lambda s.\text{done}(v),\ k) \\
\texttt{stm} \to \texttt{stm ; stm} & \mathcal{S}[\texttt{stm}_0] = \lambda(c,k).\mathcal{S}[\texttt{stm}_1](\mathcal{S}[\texttt{stm}_2](c,k),\ k) \\
 & \\
\texttt{exp} \to \texttt{(stm, exp)} & \mathcal{E}[\texttt{exp}_0] = \lambda(k,k_c).\mathcal{S}[\texttt{stm}_1](\mathcal{E}[\texttt{exp}_1](k,k_c),\ k_c) \\
\texttt{exp} \to \texttt{NUM} & \mathcal{E}[\texttt{exp}_0] = \lambda(k,k_c).k(\mathcal{N}[\texttt{NUM}]) \\
\texttt{exp} \to \texttt{ID} & \mathcal{E}[\texttt{exp}_0] = \lambda(k,k_c).\lambda s.k(s(\mathcal{I}[\texttt{ID}]))s \\
\texttt{exp} \to \texttt{exp + exp} & \mathcal{E}[\texttt{exp}_0] = \lambda(k,k_c).\mathcal{E}[\texttt{exp}_1](\lambda v_1.\mathcal{E}[\texttt{exp}_2](\lambda v_2.k(\text{plus}(v_1,v_2)),\ k_c),\ k_c) \\
\texttt{exp} \to \texttt{(exp)} & \mathcal{E}[\texttt{exp}_0] = \mathcal{E}[\texttt{exp}_1] \\
\texttt{exp} \to \texttt{CATCH exp} & \mathcal{E}[\texttt{exp}_0] = \lambda(k,k_c)\mathcal{E}[\texttt{exp}_1](k,k) \\
\texttt{exp} \to \texttt{THROW exp} & \mathcal{E}[\texttt{exp}_0] = \lambda(k,k_c)\mathcal{E}[\texttt{exp}_1](k_c,k_c)
\end{array}
$$