Structural Operational Semantics

[Specifications of operational semantics via abstract machines] "have a tendancy to pull the syntax to pieces or at any rate to wander around the syntax creating various complex symbolic structures which do not seem particularly forced by the demands of the language itself"

Gordon Plotkin, "A Structural Approach to Operational Semantics" (1981)

Structural Operational Semantics

[Specifications of operational semantics via abstract machines] "have a tendancy to pull the syntax to pieces or at any rate to wander around the syntax creating various complex symbolic structures which do not seem particularly forced by the demands of the language itself"

Gordon Plotkin, "A Structural Approach to Operational Semantics" (1981)

popularised use of rule-based inductive definitions (of various kinds of relation), where the rules are "syntax-directed"

Structural Operational Semantics

Contrasting, but related styles of SOS:

- Milner, Kahn: evaluation relations ("big-step" SOS)
- Plotkin: transition relations ("small step" SOS)
- Felleisen: transitions using evaluation-contexts ("frame stacks")

Structural Operational Semantics

Contrasting, but related styles of SOS:

- Milner, Kahn: evaluation relations ("big-step" SOS)
- Plotkin: transition relations ("small step" SOS)
- Felleisen: transitions using evaluation-contexts ("frame stacks")

We will use a fragment of ML to illustrate this [OS&PE, Appendix A] — it features:
recursively defined, higher-order, call-by-value functions
+ dynamically created mutable state.

Syntax of types (A1) & expressions (A2)

Types ty ::=

bool	booleans
int	integers
unit	unit
int ref	integer storage locations
ty $\times$ ty	pairs
ty $\rightarrow$ ty	functions

Syntax of types (A1) & expressions (A2)

Expressions e ::=

x f	value identifiers ($x, f \in Var$)
true false	booleans
if e then e else e	conditional
n	integer constant ($n \in \mathbb{Z}$)
e op e	arithmetic ($op \in \{=, +, -, \ldots\}$)
()	unit value
e, e	pair
fst e snd e	projections
fun (x : ty) -> e	function
fun f = (x : ty) -> e	recursive fn
e e	function application
let x = e in e	local definition
!e	look-up
e := e	assignment
ref e	storage creation
e == e	location equality
l	storage locations ($l \in Loc$)
e ; e	sequencing

Syntax of types (A1) & expressions (A2)

- abstract syntax trees modulo α-equivalence
  - binding forms & free variables

$$
\boxed{
\begin{aligned}
fv(\mathrm{fun}(x:ty)\to e) &\triangleq fv(e) - \{x\} \\
fv(\mathrm{fun}\ f = (x:ty)\to e) &\triangleq fv(e) - \{f, x\} \\
fv(\mathrm{let}\ x = e\ \mathrm{in}\ e') &\triangleq fv(e) \cup (fv(e') - \{x\})
\end{aligned}
}
$$

- environment-free formulation
  - so storage locations (aka "addresses") occur explicitly in expressions

$$\boxed{loc(e) \triangleq \text{locations occurring in } e}$$

Evaluation to canonical form

Canonical forms $\subseteq$ expressions :

$$
\begin{aligned}
v ::= \;& x \quad f && (x, f \in Var) \\
& \mathrm{true} \\
& \mathrm{false} \\
& n && (n \in \mathbb{Z}) \\
& () \\
& v, v \\
& \mathrm{fun}\ (x : ty) \to e \\
& \mathrm{fun}\ f = (x : ty) \to e \\
& \ell && (\ell \in Loc)
\end{aligned}
$$

ML Evaluation Semantics (simplified, environment-free form)

Evaluation relation

$\boxed{s, e \Rightarrow v, s'}$

s = initial state
e = closed expression to be evaluated
v = resulting closed canonical form
s' = final state

is inductively generated by rules *following the structure* of e, for example:

$$\frac{s, e_1 \Rightarrow v_1, s' \quad s', e_2[v_1/x] \Rightarrow v_2, s''}{s, \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \Rightarrow v_2, s''}$$

Evaluation semantics is also known as *big-step* (anon), *natural* (Kahn 1987), or *relational* (Milner) semantics.

finite function from locations to integers

ML Evaluation Semantics (simplified, environment-free form)

Evaluation relation

$\boxed{s, e \Rightarrow v, s'}$

s = initial state
e = closed expression to be evaluated
v = resulting closed canonical form
s' = final state

$fv(e) = \emptyset = fv(v)$

is inductively generated by rules *following the structure* of e, for example:

$$\frac{s, e_1 \Rightarrow v_1, s' \quad s', e_2[v_1/x] \Rightarrow v_2, s''}{s, \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \Rightarrow v_2, s''}$$

Evaluation semantics is also known as *big-step* (anon), *natural* (Kahn 1987), or *relational* (Milner) semantics.

invariant: $loc(e) \subseteq dom(s)$ & $loc(v) \subseteq dom(s')$

look-up:

$$\frac{s, e \Rightarrow l, s' \quad (l \mapsto n) \in s'}{s, !e \Rightarrow n, s'}$$

assignment:

$$\frac{s, e_1 \Rightarrow l, s' \quad s', e_2 \Rightarrow n, s''}{s, e_1 := e_2 \Rightarrow (), s''[l \mapsto n]}$$

storage creation:

$$\frac{s, e \Rightarrow n, s' \quad l \notin dom(s')}{s, \text{ref}\, e \Rightarrow l, s'[l \mapsto n]}$$

function application (I):

$$\frac{\begin{array}{c} s, e_1 \Rightarrow v_1, s' \quad s', e_2 \Rightarrow v_2, s'' \\ v_1 = \text{fun}\,(x : ty) \to e \\ s'', e[v_2/x] \Rightarrow v_3, s''' \end{array}}{s, e_1 e_2 \Rightarrow v_3\ s'''}$$

function application (II):

$$\frac{s, e_1 \Rightarrow v_1, s' \quad s', e_2 \Rightarrow v_2, s'' \quad v_1 = \mathtt{fun}\ f = (x : ty) \to e \quad s'', e[v_1/f, v_2/x] \Rightarrow v_3, s'''}{s, e_1\, e_2 \Rightarrow v_3\ s'''}$$

E.g. for $fact \triangleq \mathtt{fun}\ f = (x : int) \to \mathtt{if}\ x = 0\ \mathtt{then}\ 1\ \mathtt{else}\ x * f(x-1)$ have:

$s, fact\ 3 \Rightarrow 6, s$

'cos $s, \mathtt{if}\ 3 = 0\ \mathtt{then}\ 1\ \mathtt{else}\ 3 * fact(3-1) \Rightarrow 6, s$

'cos $s, 3 * fact(3-1) \Rightarrow 6, s$

'cos ... (etc)

Properties of evaluation relation

- if $s, e \Rightarrow v, s'$, then $dom(s) \subseteq dom(s')$
- [essentially] deterministic:
  If $s, e \Rightarrow v_1, s_1$ and $s, e \Rightarrow v_2, s_2$, then v_1, s_1 and v_2, s_2 only differ up to permutation of the freshly created locations, i.e. there is a permutation $\pi : Loc \cong Loc$ fixing $dom(s)$ [$\pi(\ell) = \ell$ for $\ell \in dom(s)$] and with

  $$\pi \cdot v_1 = v_2 \qquad \pi \cdot s_1 = s_2$$

  [Proof : . . .]

Proof strategy for showing $\Rightarrow$ is deterministic:

Consider

$$H \triangleq \{(s, e, v_1, s_1) \mid s, e \Rightarrow v_1, s_1 \ \& \ \forall v_2, s_2 .\ s, e \Rightarrow v_2, s_2 \supset \exists \pi \in Perm(Loc).\ \forall l \in dom(s).\ \pi(l) = l \ \& \ \pi \cdot v_1 = v_2 \ \& \ \pi \cdot s_1 = s_2 \}$$

and show ~~that~~ H is closed under the axioms & rules inductively defining $\Rightarrow$.

(E.g. show

$(s, e, n_1, s_1) \in H \ \& \ l_1 \notin dom(s) \supset (s, ref\,e, l_1, s[l_1 \mapsto n_1]) \in H$

etc.)

Properties of evaluation relation

- non-termination: given s, e (with $loc(e) \subseteq dom(s)$), there can be <u>no</u> v, s' with $s, e \Rightarrow v, s'$.

The non-termination of

$(\mathrm{fun}\ f = (x : unit) \to f\,x)(\,)$ [divergence]

is qualitatively different from that for

$3(\,)$ [type error]

– we use a type system to statically check for the latter kind of non-termination...