Opal Semantics: Technical Report

Alex Renda, Harrison Goldstein, Sarah Bird, Chris Quirk, Adrian Sampson

January 31, 2018

1 Description

Tiny turing-incomplete language based on Imp with sexps:

2 Grammar

$\langle Com \rangle$ ::= `skip`
| $\langle Com \rangle$; $\langle Com \rangle$
| $\langle Node \rangle$.$\langle Var \rangle$:= $\langle Sexp \rangle$
| `if` $\langle Bool \rangle$ `then` {$\langle Com \rangle$} `else` {$\langle Com \rangle$}
| `with` $\langle Node \rangle$ `do` {$\langle Com \rangle$}
| `at` $\langle Node \rangle$ `do` {$\langle Com \rangle$}
| $\langle WorldVar \rangle$:= `hyp` {c}
| `commit` $\langle WorldVar \rangle$

$\langle Sexp \rangle$ ::= $\varnothing$
| $\langle Node \rangle$.$\langle Var \rangle$
| $\langle World \rangle$.$\langle Node \rangle$.$\langle Var \rangle$
| ($\langle Sexp \rangle$. $\langle Sexp \rangle$)

$\langle Bool \rangle$ ::= $\langle Sexp \rangle$ = $\langle Sexp \rangle$
| $\langle Sexp \rangle$ $\in$ $\langle Sexp \rangle$
| $\langle Bool \rangle$ $\wedge$ $\langle Bool \rangle$
| $\langle Bool \rangle$ $\vee$ $\langle Bool \rangle$
| `true`
| `false`

2.1 Values

$\langle SexpValue \rangle$ ::= $\varnothing$
| ($\langle SexpValue \rangle$. $\langle SexpValue \rangle$)

$\langle BoolValue \rangle$ ::= `true`
| `false`

3 Typing Judgement

Well-formedness conditions:

- All node accesses are permissioned via a `with` block
- All variable accesses are defined (if they access a world, that world is defined and not committed)
- All committed worlds are defined
- Worlds are committed at most once

3.1 Terminology

Σ :	$2^{\langle Node \rangle \times \langle Var \rangle}$	Set of variables in scope
Ω :	$2^{\langle World \rangle}$	Set of worlds in scope
Π :	$2^{\langle Node \rangle}$	Set of nodes giving permission to access variables

3.2 Commands

Commands follow an affine typing scheme for worlds, wherein worlds can be used *at most* once. This is achieved by a conservative judgement over commands: declaring a hypothetical world puts it into scope, committing it takes it out of scope, and if statements take the intersection of available worlds after each branch.

The typing judgement for commands is a relation $R \subseteq \Sigma \times \Omega \times \Pi \times \langle Com \rangle \times \Omega$, where intuitively the first three members are the state before the command is executed, and the final Ω is the conservative judgement of the set of available worlds after the command is executed.

$$\frac{}{\Sigma, \Omega, \Pi \vdash \texttt{skip} : \Omega} \text{ T-SKIP}$$

$$\frac{\Sigma, \Omega, \Pi \vdash c_1 : \Omega' \qquad \Sigma, \Omega', \Pi \vdash c_2 : \Omega''}{\Sigma, \Omega, \Pi \vdash c_1 ; c_2 : \Omega''} \text{ T-SEQ}$$

$$\frac{\Sigma, \Omega, \Pi \vdash b : \text{BOOL} \qquad \Sigma, \Omega, \Pi \vdash c_1 : \Omega' \qquad \Sigma, \Omega, \Pi \vdash c_2 : \Omega''}{\Sigma, \Omega, \Pi \vdash \texttt{if } b \texttt{ then } \{c_1\} \texttt{ else } \{c_2\} : \Omega' \cap \Omega''} \text{ T-IF}$$

$$\frac{\Sigma, \varnothing, \Pi \vdash c : \Omega'}{\Sigma, \Omega, \Pi \vdash u \texttt{ := } \texttt{hyp } \{c\} : \Omega \cup \{u\}} \text{ T-ASSIGNWORLD}$$

$$\frac{u \in \Omega}{\Sigma, \Omega, \Pi \vdash \texttt{commit}\ u : \Omega \setminus \{u\}}\ \textsc{T-CommitWorld}$$

$$\frac{\Sigma, \Omega, \Pi \cup \{n\} \vdash c : \Omega'}{\Sigma, \Omega, \Pi \vdash \texttt{with}\ n\ \texttt{do}\ \{c\} : \Omega'}\ \textsc{T-With}$$

$$\frac{\Sigma, \Omega, \Pi \vdash c : \Omega'}{\Sigma, \Omega, \Pi \vdash \texttt{at}\ n\ \texttt{do}\ \{c\} : \Omega'}\ \textsc{T-At}$$

3.3 Booleans

$$\frac{}{\Sigma, \Omega, \Pi \vdash \texttt{true} : \textsc{Bool}}\ \textsc{T-True}$$

$$\frac{}{\Sigma, \Omega, \Pi \vdash \texttt{false} : \textsc{Bool}}\ \textsc{T-False}$$

$$\frac{\Sigma, \Omega, \Pi \vdash b_1 : \textsc{Bool} \qquad \Sigma, \Omega, \Pi \vdash b_2 : \textsc{Bool}}{\Sigma, \Omega, \Pi \vdash b_1 \wedge b_2 : \textsc{Bool}}\ \textsc{T-Conj}$$

$$\frac{\Sigma, \Omega, \Pi \vdash b_1 : \textsc{Bool} \qquad \Sigma, \Omega, \Pi \vdash b_2 : \textsc{Bool}}{\Sigma, \Omega, \Pi \vdash b_1 \vee b_2 : \textsc{Bool}}\ \textsc{T-Disj}$$

$$\frac{\Sigma, \Omega, \Pi \vdash s_1 : \textsc{Sexp} \qquad \Sigma, \Omega, \Pi \vdash s_2 : \textsc{Sexp}}{\Sigma, \Omega, \Pi \vdash s_1 = s_2 : \textsc{Bool}}\ \textsc{T-Eq}$$

$$\frac{\Sigma, \Omega, \Pi \vdash s_1 : \textsc{Sexp} \qquad \Sigma, \Omega, \Pi \vdash s_2 : \textsc{Sexp}}{\Sigma, \Omega, \Pi \vdash s_1 \in s_2 : \textsc{Bool}}\ \textsc{T-Mem}$$

3.4 Sexps

$$\frac{}{\Sigma, \Omega, \Pi \vdash \varnothing : \textsc{Sexp}}\ \textsc{T-EmptySexp}$$

$$\frac{n \in \Pi \qquad (n, v) \in \Sigma}{\Sigma, \Omega, \Pi \vdash n.v : \textsc{Sexp}}\ \textsc{T-Variable}$$

$$\frac{u \in \Omega \qquad n \in \Pi \qquad (n, v) \in \Sigma}{\Sigma, \Omega, \Pi \vdash u.n.v : \textsc{Sexp}}\ \textsc{T-Weight}$$

$$\frac{\Sigma, Omega, \Pi \vdash s_1 : \textsc{Sexp} \qquad \Sigma, Omega, \Pi \vdash s_2 : \textsc{Sexp}}{\Sigma, Omega, \Pi \vdash (s_1.s_2) : \textsc{Sexp}}\ \textsc{T-Cons}$$

4 Big Step Semantics

4.1 Terminology

σ :	$\langle \mathit{Node} \rangle \times \langle \mathit{Var} \rangle \rightharpoonup \langle \mathit{SexpValue} \rangle$	Partial function representing each node's store
ω :	$\langle \mathit{World} \rangle \rightharpoonup \sigma$	Partial function representing each executed world's final store
π :	$2^{\langle \mathit{Node} \rangle}$	Current authorized set of principals
ρ :	$\langle \mathit{Node} \rangle$	Current execution location
μ :	$(\langle \mathit{Sexp} \rangle \times \langle \mathit{Sexp} \rangle) \rightharpoonup \langle \mathit{Sexp} \rangle$	Partial(!) function which merges two divergent data structures
$\Downarrow_{\langle \mathit{Com} \rangle}$:	$(\langle \mathit{Com} \rangle \times \sigma \times \omega \times \pi \times \rho \times \mu) \rightharpoonup (\sigma \times \omega)$	Commands step to a new store and new set of hypothetical worlds
$\Downarrow_{\langle \mathit{Sexp} \rangle}$:	$(\langle \mathit{Sexp} \rangle \times \sigma \times \pi) \rightharpoonup \langle \mathit{SexpValue} \rangle$	Sexps step to a value if they can be evaluated
$\Downarrow_{\langle \mathit{Bool} \rangle}$:	$(\langle \mathit{Bool} \rangle \times \sigma \times \pi) \rightharpoonup \langle \mathit{BoolValue} \rangle$	Bools step to a value if they can be evaluated

4.2 Basic commands

$$\frac{}{\langle \texttt{skip}, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma, \omega \rangle} \text{ E-S}\textsc{kip}$$

$$\frac{\langle c_1, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle \qquad \langle c_2, \sigma', \omega', \pi, \rho, \mu \rangle \Downarrow \langle \sigma'', \omega'' \rangle}{\langle c_1;\ c_2, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma'', \omega'' \rangle} \text{ E-S}\textsc{eq}$$

$$\frac{\langle b, \sigma, \pi \rangle \Downarrow \texttt{true} \qquad \langle c_1, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle}{\langle \texttt{if } b \texttt{ then } \{c_1\} \texttt{ else } \{c_2\}, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle} \text{ E-I}\textsc{f}\text{-T}\textsc{rue}$$

$$\frac{\langle b, \sigma, \pi \rangle \Downarrow \texttt{false} \qquad \langle c_2, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle}{\langle \texttt{if } b \texttt{ then } \{c_1\} \texttt{ else } \{c_2\}, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle} \text{ E-I}\textsc{f}\text{-F}\textsc{alse}$$

4.3 Distribution commands

In this semantics, AT is effectively a noop/passthrough, resulting in behavior that would be identical with or without it (modulo endorsement taking location into account).

WITH is also a passthrough in a similar regard: its runtime effects consist of asking the specified user for permission to run the specified command on the current machine (represented by $\langle n, \rho, c \rangle \checkmark$ in the semantics – the details of this function are implementation dependent).

$$\frac{\langle c, \sigma, \omega, \pi, n, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle}{\langle \texttt{at}\ n\ \texttt{do}\ \{c\}, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle}\ \text{E-AT}$$

$$\frac{\langle n, \rho, c \rangle \checkmark \qquad \langle c, \sigma, \omega, \pi \cup \{n\}, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle}{\langle \texttt{with}\ n\ \texttt{do}\ \{c\}, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma', \omega' \rangle}\ \text{E-WITH}$$

4.4 Hypothetical commands

$$\frac{\langle c, \sigma, \varnothing, \pi, \rho, \mu \rangle \Downarrow \langle \sigma_{\text{hyp}}, \omega_{\text{hyp}} \rangle}{\langle u\ :=\ \texttt{hyp}\ \{c\}, \sigma, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma, \omega[u \mapsto \sigma_{\text{hyp}}] \rangle}\ \text{E-HYP}$$

$$\frac{\omega[u] = \sigma_{\text{hyp}} \qquad \forall v \in \sigma_{\text{hyp}}.\ \sigma_{\text{merge}}[v] = \mu(\sigma_{\text{curr}}[v], \sigma_{\text{hyp}}[v]) \qquad \forall v \notin \sigma_{\text{hyp}}.\nexists s.\sigma_{\text{merge}}[v] = s}{\langle \texttt{commit}\ u, \sigma_{\text{curr}}, \omega, \pi, \rho, \mu \rangle \Downarrow \langle \sigma_{\text{merge}}; \sigma_{\text{curr}}, \omega \rangle}\ \text{E-COMMIT}$$

4.5 Sexps

$$\frac{}{\langle \varnothing, \sigma, \omega, \pi \rangle \Downarrow \varnothing}\ \text{E-EMPTYSEXP}$$

$$\frac{\langle s_1, \sigma, \omega, \pi \rangle \Downarrow s_1' \qquad \langle s_2, \sigma, \omega, \pi \rangle \Downarrow s_2'}{\langle (s_1.s_2), \sigma, \omega, \pi \rangle \Downarrow (s_1'.s_2')}\ \text{E-CONS}$$

$$\frac{n \in \pi \qquad \sigma[n, v] = s}{\langle n.v, \sigma, \omega, \pi \rangle \Downarrow s}\ \text{E-VAR}$$

$$\frac{n \in \pi \qquad \omega[u] = \sigma_{\text{hyp}} \qquad \sigma_{\text{hyp}}[n, v] = s}{\langle u.n.v, \sigma, \omega, \pi \rangle \Downarrow s}\ \text{E-WEIGHT}$$

4.6 Bools

$$\frac{}{\langle \texttt{true}, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}} \textsc{E-True}$$

$$\frac{}{\langle \texttt{false}, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}} \textsc{E-False}$$

$$\frac{\langle b_1, \sigma, \omega, \pi \rangle \Downarrow \texttt{true} \qquad \langle b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}}{\langle b_1 \wedge b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}} \textsc{E-AndTrue}$$

$$\frac{\langle b_1, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}}{\langle b_1 \wedge b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}} \textsc{E-AndFalseL}$$

$$\frac{\langle b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}}{\langle b_1 \wedge b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}} \textsc{E-AndFalseR}$$

$$\frac{\langle b_1, \sigma, \omega, \pi \rangle \Downarrow \texttt{false} \qquad \langle b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}}{\langle b_1 \vee b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}} \textsc{E-OrFalse}$$

$$\frac{\langle b_1, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}}{\langle b_1 \vee b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}} \textsc{E-OrTrueL}$$

$$\frac{\langle b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}}{\langle b_1 \vee b_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}} \textsc{E-OrTrueR}$$

$$\frac{\langle s_1, \sigma, \omega, \pi \rangle \Downarrow \varnothing \qquad \langle s_2, \sigma, \omega, \pi \rangle \Downarrow \varnothing}{\langle s_1 = s_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{true}} \textsc{E-EqTrue}$$

$$\frac{\langle s_1, \sigma, \omega, \pi \rangle \Downarrow (s_{v11}.s_{v12}) \qquad \langle s_2, \sigma, \omega, \pi \rangle \Downarrow \varnothing}{\langle s_1 = s_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}} \textsc{E-EqFalseL}$$

$$\frac{\langle s_1, \sigma, \omega, \pi \rangle \Downarrow \varnothing \qquad \langle s_2, \sigma, \omega, \pi \rangle \Downarrow (s_{v21}.s_{v22})}{\langle s_1 = s_2, \sigma, \omega, \pi \rangle \Downarrow \texttt{false}} \textsc{E-EqFalseR}$$

$$\frac{\langle s_1, \sigma, \omega, \pi\rangle \Downarrow (s_{v11}.s_{v12}) \qquad \langle s_2, \sigma, \omega, \pi\rangle \Downarrow (s_{v21}.s_{v22}) \qquad \langle s_{v11} = s_{v21} \wedge s_{v12} = s_{v22}, \sigma, \pi\rangle \Downarrow b}{\langle s_1 = s_2, \sigma, \omega, \pi\rangle \Downarrow b} \text{ E-EqProp}$$

$$\frac{\langle s_2, \sigma, \omega, \pi\rangle \Downarrow \varnothing}{\langle s_1 \in s_2, \sigma, \omega, \pi\rangle \Downarrow \texttt{false}} \text{ E-MemFalse}$$

$$\frac{\langle s_2, \sigma, \omega, \pi\rangle \Downarrow (s_{v21}.s_{v22}) \qquad \langle s_1 = s_{v21} \vee s_1 \in s_{v22}, \sigma, \omega, \pi\rangle \Downarrow b}{\langle s_1 \in s_2, \sigma, \omega, \pi\rangle \Downarrow b} \text{ E-MemProp}$$

EqProp and MemProp are well-founded proof trees since the sexp step is idempotent, and the two props decrease on size of sexp, so the maximum depth of the proof tree is proportional to the maximum depth of the sexp values