ESCAPE ANALYSIS

TOMO KAZAHAYA AND REBECCA CHEN

1. Definitions

1.1. **Intuition.** In **let** $x = v$ **in** e, the value v escapes if it is used outside e. We will consider the lambda calculus:

$$\begin{aligned} e \in Exp &::= x \mid \lambda x.\ e \mid e\ e \\ v \in Val &::= \lambda x.\ e \\ x \in Var & \quad \text{a set of identifiers} \end{aligned}$$

in which the equivalent expression is $(\lambda x.\ e)\ v$. So v - an abstraction - escapes if it is applied outside e.

1.2. **Modifying the time-stamped $CESK^*$ machine.** This section draws heavily from [1, 2].

A state of the time-stamped $CESK^*$ machine consists of an expression, an environment that maps variables to addresses, a store that maps addresses to closures or continuations, a continuation pointer, and a time.

$$\begin{aligned} \zeta \in \Sigma &= Exp \times Env \times Store \times Addr \times Time \\ \rho \in Env &= Var \to Addr \\ \sigma \in Store &= Addr \to Val \times Exp + Kont \\ \kappa \in Kont &::= \mathbf{mt} \mid \langle \mathbf{ar}, e, \rho, a \rangle \mid \langle \mathbf{fn}, v, \rho, a \rangle \mid \\ a \in Addr & \quad \text{an infinite set} \\ t \in Time & \quad \text{an infinite set} \end{aligned}$$

Functions *alloc* and *tick* return a fresh address in the store and the next time, respectively:

$$alloc : \Sigma \to Addr \quad tick : \Sigma \to Time$$

For an expression e, the initial state is given by the *inj* function:

$$inj(e) = \langle e, \emptyset, [a_0 \mapsto \mathbf{mt}], a_0, t_0 \rangle.$$

Figure 1 shows the transition rules. Rule 4, in which an abstraction is applied, is the crucial one. We need to check whether $\lambda x.\ e$ has escaped as well as somehow store the information that v escapes if it

$\zeta \longrightarrow \zeta'$, where $a' = alloc(\zeta), t' = tick(\zeta)$	
1	$\langle x, \rho, \sigma, a, t\rangle \longrightarrow$
	$\langle v, \rho_v, \sigma, a, t'\rangle$ where $\langle v, \rho_v\rangle = \sigma(\rho(x))$
2	$\langle e_1 e_2, \rho, \sigma, a, t\rangle \longrightarrow$
	$\langle e_1, \rho, \sigma[a' \mapsto \langle \mathbf{ar}, e_2, \rho, a\rangle], a', t'\rangle$
	$\langle v, \rho, \sigma, a, t\rangle \longrightarrow$
3	if $\sigma(a) = \langle \mathbf{ar}, e, \rho_\kappa, a_\kappa\rangle$
	$\langle e, \rho_\kappa, \sigma[a' \mapsto \langle \mathbf{fn}, v, \rho, a_\kappa\rangle], a', t'\rangle$
4	if $\sigma(a) = \langle \mathbf{fn}, \lambda x.\ e, \rho_\kappa, a_\kappa\rangle$
	$\langle e, \rho_\kappa[x \mapsto a'], \sigma[a' \mapsto \langle v, \rho\rangle], a_\kappa, t'\rangle$

FIGURE 1. Transitions of the time-stamped $CESK^*$ machine

is applied after the evaluation of e has finished. What we do is tag v with a_κ, the address of the continuation that we will go to when we are done evaluating e:

$$Val ::= \lambda x.\ e \mid (\lambda x.\ e)^a$$

The new transition rules are in Figure 2. In Rule 4a, v is tagged with

$\zeta \longrightarrow \zeta'$, where $a' = alloc(\zeta), t' = tick(\zeta)$	
1	$\langle x, \rho, \sigma, a, t\rangle \longrightarrow$
	$\langle v, \rho_v, \sigma, a, t'\rangle$ where $\langle v, \rho_v\rangle = \sigma(\rho(x))$
2	$\langle e_1 e_2, \rho, \sigma, a, t\rangle \longrightarrow$
	$\langle e_1, \rho, \sigma[a' \mapsto \langle \mathbf{ar}, e_2, \rho, a\rangle], a', t'\rangle$
	$\langle v, \rho, \sigma, a, t\rangle \longrightarrow$
3	if $\sigma(a) = \langle \mathbf{ar}, e, \rho_\kappa, a_\kappa\rangle$
	$\langle e, \rho_\kappa, \sigma[a' \mapsto \langle \mathbf{fn}, v, \rho, a_\kappa\rangle], a', t'\rangle$
4	if $\sigma(a) = \langle \mathbf{fn}, v_\kappa, \rho_\kappa, a_\kappa\rangle$, $v_\kappa = \lambda x_\kappa.\ e_\kappa$ or $(\lambda x_\kappa.\ e_\kappa)^{a_{\mathrm{abs}}}$
$4a$	if $v = \lambda x.\ e$
	$\langle e_\kappa, \rho_\kappa[x_\kappa \mapsto a'], \sigma[a' \mapsto \langle v^{a_\kappa}, \rho\rangle], a_\kappa, t'\rangle$
$4b$	if $v = (\lambda x.\ e)^{a_v}$
	$\langle e_\kappa, \rho_\kappa[x_\kappa \mapsto a'], \sigma[a' \mapsto \langle v, \rho\rangle], a_\kappa, t'\rangle$

FIGURE 2. Transitions of the modified time-stamped $CESK^*$ machine

the continuation pointer when it is put into the store for the first time. Rule 4b preserves the tag on a v that was previously put into the store and read out again. In both rules, the abstraction v_k that is applied escapes if it has a tag a_{abs} that is not reachable in σ from a_κ, where reachability is defined as follows:

Let $a, a' \in \mathit{Addr}$ and $\sigma \in \mathit{Store}$.

$$a \longrightarrow_\sigma a' := \text{“}\sigma(a) = \langle \mathbf{ar}, \_, \_, a' \rangle \text{ or } \sigma(a) = \langle \mathbf{fn}, \_, \_, a' \rangle\text{”}$$

$$\text{“}a' \text{ is reachable in } \sigma \text{ from } a\text{”} := a \longrightarrow_\sigma^* a'$$

Note that any address is reachable from itself.

References

[1] Abstract Register Machines. Lecture notes at `http://www.seas.harvard.edu/courses/cs152/2015sp/` (accessed October 2015).

[2] D. Van Horn and M. Might. Abstracting Abstract Machines. International Conference on Functional Programming 2010, 51—62 (September 2010).