Mutable State

Yu Zhang

Course web site:
http://staff.ustc.edu.cn/~yuzhang/tpl

References

- PFPL
  - Chapter 34 Modernized Algol
  - Chapter 35 Assignable References

Basic Commands

- Commands
  - Act on *assignables* by retrieving and altering their contents
- ***Pure*** expressions and ***impure*** commands
  - Pure: not depend on assignables
  - Impure: depend on assignables
- Syntax of MA

Typ	τ	::=	cmd	cmd
Exp	e	::=	cmd(m)	cmd m
Cmd	m	::=	ret(e)	ret e
			bnd(e; $x.m$)	bnd $x \leftarrow e$; m
			dcl(e; $a.m$)	dcl $a := e$ in m
			get[a]	@ a
			set[a](e)	$a := e$

command
encapsulation
return
sequence
new assignable
fetch
assign

未求值的命令m

返回e的值而不影响可赋值对象

声明一个可赋值对象

求e的值并封装成命令，执行该命令以作用于m中的可赋值对象，将该值代换m中的x

Basic Commands

- A MA Program

```
proc (x:nat) {
  dcl r := 1 in
  dcl a := x in
  { while ( @ a ) {
      y ← @ r
    ; z ← @ a
    ; r := (x-z+1)× y
    ; a := z-1
    }
    ; x ← @ r
    ; ret x
  }
}
```

Typ	τ	::=	cmd	cmd	command
Exp	e	::=	cmd(m)	cmd m	encapsulation
Cmd	m	::=	ret(e)	ret e	return
			bnd(e; $x.m$)	bnd $x \leftarrow e$; m	sequence
			dcl(e; $a.m$)	dcl $a := e$ in m	new assignable
			get[a]	@ a	fetch
			set[a](e)	$a := e$	assign

Statics of MA

- Judgements: Σ-a finite set of assignables
  - Expression typing: $\Gamma \vdash_\Sigma e : \tau$
  - Command formation: $\Gamma \vdash_\Sigma m\ ok$
- Rules

$$\frac{\Gamma \vdash_\Sigma m\ \mathtt{ok}}{\Gamma \vdash_\Sigma \mathtt{cmd}(m) : \mathtt{cmd}}$$

$$\frac{\Gamma \vdash_\Sigma e : \mathtt{nat}}{\Gamma \vdash_\Sigma \mathtt{ret}(e)\ \mathtt{ok}} \qquad \frac{}{\Gamma \vdash_{\Sigma,a} \mathtt{get}[a]\ \mathtt{ok}}$$

$$\frac{\Gamma \vdash_\Sigma e : \mathtt{cmd} \quad \Gamma, x : \mathtt{nat} \vdash_\Sigma m\ \mathtt{ok}}{\Gamma \vdash_\Sigma \mathtt{bnd}(e; x.m)\ \mathtt{ok}} \qquad \frac{\Gamma \vdash_{\Sigma,a} e : \mathtt{nat}}{\Gamma \vdash_{\Sigma,a} \mathtt{set}[a](e)\ \mathtt{ok}} \quad (34.1)$$

$$\frac{\Gamma \vdash_\Sigma e : \mathtt{nat} \quad \Gamma \vdash_{\Sigma,a} m\ \mathtt{ok}}{\Gamma \vdash_\Sigma \mathtt{dcl}(e; a.m)\ \mathtt{ok}}$$

Dynamics of MA

- Judgements
  - $e\ \mathrm{val}_\Sigma$: e is a value relative to Σ
  - $e \mapsto_\Sigma e'$: e steps to e'
  - $m \parallel \mu\ \mathrm{final}_\Sigma$: the state $m \parallel \mu$ is complete
    - μ : a memory mapping assignables to values
  - $m \parallel \mu \mapsto_\Sigma m' \parallel \mu'$: $m \parallel \mu$ steps to $m' \parallel \mu'$
- Rules

$$\frac{}{\mathtt{cmd}(m)\ \mathrm{val}_\Sigma}$$

$$\frac{e\ \mathrm{val}_\Sigma}{\mathtt{ret}(e) \parallel \mu\ \mathrm{final}_\Sigma}$$

$$\frac{e \underset{\Sigma}{\mapsto} e'}{\mathtt{ret}(e) \parallel \mu \underset{\Sigma}{\mapsto} \mathtt{ret}(e') \parallel \mu}$$

Dynamics of MA

- Rules

$$\frac{e \mapsto_{\Sigma} e'}{\mathtt{bnd}(e;x.m) \parallel \mu \mapsto_{\Sigma} \mathtt{bnd}(e';x.m) \parallel \mu}$$

$$\frac{e \ \mathrm{val}_{\Sigma}}{\mathtt{bnd}(\mathtt{cmd}(\mathtt{ret}(e));x.m) \parallel \mu \mapsto_{\Sigma} [e/x]m \parallel \mu}$$

$$\frac{m_1 \parallel \mu \mapsto_{\Sigma} m_1' \parallel \mu'}{\mathtt{bnd}(\mathtt{cmd}(m_1);x.m_2) \parallel \mu \mapsto_{\Sigma} \mathtt{bnd}(\mathtt{cmd}(m_1');x.m_2) \parallel \mu'}$$

$$\overline{\mathtt{get}[a] \parallel \mu \otimes a \hookrightarrow e \longmapsto_{\Sigma,a} \mathtt{ret}(e) \parallel \mu \otimes a \hookrightarrow e}$$

define the concept of *block structure*

Declarations adhere to the *stack discipline*

$$\frac{e \mapsto_{\Sigma,a} e'}{\mathtt{set}[a](e) \parallel \mu \longmapsto_{\Sigma,a} \mathtt{set}[a](e') \parallel \mu}$$

$$\frac{e \ \mathrm{val}_{\Sigma,a}}{\mathtt{set}[a](e) \parallel \mu \otimes a \hookrightarrow \_ \longmapsto_{\Sigma,a} \mathtt{ret}(e) \parallel \mu \otimes a \hookrightarrow e}$$

$$\frac{e \mapsto_{\Sigma} e'}{\mathtt{dcl}(e;a.m) \parallel \mu \mapsto_{\Sigma} \mathtt{dcl}(e';a.m) \parallel \mu}$$

$$\frac{e \ \mathrm{val}_{\Sigma} \quad m \parallel \mu \otimes a \hookrightarrow e \longmapsto_{\Sigma,a} m' \parallel \mu' \otimes a \hookrightarrow e'}{\mathtt{dcl}(e;a.m) \parallel \mu \mapsto_{\Sigma} \mathtt{dcl}(e';a.m') \parallel \mu'}$$

$$\frac{e \ \mathrm{val}_{\Sigma} \quad e' \ \mathrm{val}_{\Sigma,a}}{\mathtt{dcl}(e;a.\mathtt{ret}(e')) \parallel \mu \mapsto_{\Sigma} \mathtt{ret}(e') \parallel \mu}$$

Safety of MA

- Judgements
  - $m \parallel \mu \ \mathrm{ok}_{\Sigma}$ is defined by the rule

$$\frac{\vdash_{\Sigma} m \ \mathrm{ok} \quad \mu : \Sigma}{m \parallel \mu \ \mathrm{ok}_{\Sigma}} \tag{34.4}$$

$$\frac{\forall a \in \Sigma \quad \exists e \quad \mu(a) = e \text{ and } e \ \mathrm{val}_{\emptyset} \text{ and } \vdash_{\emptyset} e : \mathrm{nat}}{\mu : \Sigma} \tag{34.5}$$

- Preservation
  - If $e \mapsto_{\Sigma} e'$and $\vdash_{\Sigma} e : \tau$, then $\vdash_{\Sigma} e' : \tau$
  - If $m \parallel \mu \mapsto_{\Sigma} m' \parallel \mu'$, with $\vdash_{\Sigma} m \ ok$ and $\mu : \Sigma$, then $\vdash_{\Sigma} m' ok$ and $\mu' : \Sigma$
- Progress
  - If $\vdash_{\Sigma} e : \tau$, then either $e \ \mathrm{val}_{\Sigma}$ or there exists e' such that $e \mapsto_{\Sigma} e'$
  - If $\vdash_{\Sigma} m \ ok$ and $\mu : \Sigma$, then either $m \parallel \mu \ \mathrm{final}_{\Sigma}$ or $m \parallel \mu \mapsto_{\Sigma} m' \parallel \mu'$ for some μ' and m'

Assignable References: RMA

- A reference to an assignable a is a value, $\&a$
- Reference types are compatible with
  - a scoped assignable
  - a scope-free allocation of assignable
- Scoped Assignables

Typ	τ	::=	$\mathtt{ref}(\tau)$	τ ref
Exp	e	::=	$\mathtt{ref}[a]$	$\&a$
Cmd	m	::=	$\mathtt{getref}(e)$	$*e$
			$\mathtt{setref}(e_1;e_2)$	$e_1 *= e_2$

assignable
reference
contents
update

Scoped Assignables

- Statics

$$\overline{\Gamma \vdash_{\Sigma,a\sim\tau} \mathtt{ref}[a] : \mathtt{ref}(\tau)}$$

$$\frac{\Gamma \vdash_{\Sigma} e : \mathtt{ref}(\tau)}{\Gamma \vdash_{\Sigma} \mathtt{getref}(e) \div \tau}$$

$$\frac{\Gamma \vdash_{\Sigma} e_1 : \mathtt{ref}(\tau) \quad \Gamma \vdash_{\Sigma} e_2 : \tau}{\Gamma \vdash_{\Sigma} \mathtt{setref}(e_1;e_2) \div \tau}$$

$\Gamma \vdash_{\Sigma} m \div \tau$
m is a well-formed command returning a value of type τ

- Dynamics

$$\overline{\mathtt{ref}[a] \ \mathrm{val}_{\Sigma,a\sim\tau}}$$

$$\frac{e \mapsto_{\Sigma} e'}{\mathtt{getref}(e) \parallel \mu \mapsto_{\Sigma} \mathtt{getref}(e') \parallel \mu}$$

$$\overline{\mathtt{getref}(\mathtt{ref}[a]) \parallel \mu \longmapsto_{\Sigma,a\sim\tau} \mathtt{get}[a] \parallel \mu}$$

$$\frac{e_1 \mapsto_{\Sigma} e_1'}{\mathtt{setref}(e_1;e_2) \parallel \mu \mapsto_{\Sigma} \mathtt{setref}(e_1';e_2) \parallel \mu}$$

$$\overline{\mathtt{setref}(\mathtt{ref}[a];e) \parallel \mu \longmapsto_{\Sigma,a\sim\tau} \mathtt{set}[a](e) \parallel \mu}$$

Free Assignables

- Scope-free assignables: Heap allocation
- Dynamics: allocation of assignables persists across transitions
  - Transition judgement $\nu\,\Sigma\,\{m \parallel \mu\} \longmapsto \nu\,\Sigma'\,\{m' \parallel \mu'\}$.
  - command $\mathrm{newref}[\tau](e)$ defined by

$$\mathtt{dcl}\ a := e\ \mathtt{in}\ \mathtt{ret}(\&a). \tag{35.5}$$

$$\frac{\Gamma \vdash_{\Sigma} e : \tau}{\Gamma \vdash_{\Sigma} \mathtt{newref}[\tau](e) \div \mathtt{ref}(\tau)} \tag{35.6}$$

$$\frac{e \mapsto_{\Sigma} e'}{\nu\,\Sigma\,\{\mathtt{newref}[\tau](e) \parallel \mu\} \longmapsto \nu\,\Sigma\,\{\mathtt{newref}[\tau](e') \parallel \mu\}} \tag{35.7a}$$

$$\frac{e \ \mathrm{val}_{\Sigma}}{\nu\,\Sigma\,\{\mathtt{newref}[\tau](e) \parallel \mu\} \longmapsto \nu\,\Sigma, a\sim\tau\,\{\mathtt{ret}(\mathtt{ref}[a]) \parallel \mu \otimes a \hookrightarrow e\}} \tag{35.7b}$$