

# Relational semantics of store operations

Lennart Beringer

join work with N. Benton, A. Kennedy, and M. Hofmann

Lehrstuhl für Theoretische Informatik  
Ludwig-Maximilians-Universität München

Microsoft Research, Cambridge

EffTT Tallinn, 13 December 2007

# Motivation

## Question

When is the program equivalence

$$C; \text{if } B \text{ then } C' \text{ else } C'' = \text{if } B \text{ then } C; C' \text{ else } C; C''$$

legal in the simple language of while-commands with boolean expressions?

## Answer

Whenever  $B$  does not read any variable that  $C$  may write.

Goal of this work:

- justify program equivalences in the presence of side effects (access to variables)

# Motivation

## Question

When is the program equivalence

$$C; \text{if } B \text{ then } C' \text{ else } C'' = \text{if } B \text{ then } C; C' \text{ else } C; C''$$

legal in the simple language of while-commands with boolean expressions?

## Answer

Whenever  $B$  does not read any variable that  $C$  may write.

Goal of this work:

- justify program equivalences in the presence of side effects (access to variables)

# Method

- consider type and effect systems that (over)-approximate variable access, i.e. guarantee that certain variables will *not* be read/modified.
- Interpretations of such an assertion:
  - Intensionally/operationally: intuitively clear (execution traces)
  - Extensionally?
- Answer: “preservation of certain binary (store) relations”

## Example for while - programs over 2 variables $X$ and $Y$

- $C$  does not observably write to  $X$ :  $C$  preserves all relations  $R \subseteq \Delta_X$
- $C$  does not read from  $X$ :  $C$  preserves all relations  $R \supseteq \Delta_X$
- $C$  observably neither reads from nor writes to  $X$ :  $C$  preserves all relations  $R$

# This talk

- Higher-order language with dynamically allocated integer references
- Region-based type and effect system, including “masking rule”, for tracking reading, writing and allocation
- Interpretation: binary store relations, parametrized by region-indexed partial bijections on locations
- Transformations: duplicated, commuting, and dead computations, pure lambda hoist
- Details in PPDP'07 paper

# Language

- Monadically typed, CBV  $\lambda$  calculus, plus expressions  $\text{read}(V)$ ,  $\text{write}(V_1, V_2)$  and  $\text{ref}(V)$ .
- Basic type system: value types

$$A ::= \text{unit} \mid \text{int} \mid \text{bool} \mid \text{ref} \mid A \times B \mid A \rightarrow TB$$

plus computation types  $TA$ . Example rules:

$$\frac{\Gamma \vdash V : \text{ref}}{\Gamma \vdash \text{read}(V) : T\text{int}} \quad \frac{\Gamma \vdash V_1 : \text{ref} \quad \Gamma \vdash V_2 : \text{int}}{\Gamma \vdash \text{write}(V_1, V_2) : T\text{unit}} \quad \frac{\Gamma \vdash V : \text{int}}{\Gamma \vdash \text{ref}(V) : T\text{ref}}$$

# Denotational semantics

- States  $s \in \mathbb{S}$  defined over abstract set  $\mathbb{L}$  of locations, with usual operations (eg. implemented as finite map)
- Types are interpreted in the category of sets and functions (no recursion), e.g.

$$\begin{array}{ll} \llbracket \text{int} \rrbracket = \mathbb{Z} & \llbracket \text{ref} \rrbracket = \mathbb{L} \\ \llbracket A \rightarrow TB \rrbracket = \llbracket A \rrbracket \rightarrow \llbracket TB \rrbracket & \llbracket TA \rrbracket = \mathbb{S} \rightarrow \mathbb{S} \times \llbracket A \rrbracket \end{array}$$

# Effect system

- Assume infinite set  $Regs$  of region identifiers, ranged over by  $r$
- Effects  $\varepsilon$ : finite sets of *primitive effects*  $al_r, wr_r, rd_r$
- Refined value types

$$X, Y := \text{unit} \mid \text{int} \mid \text{bool} \mid \text{ref}_r \mid X \times Y \mid X \rightarrow T_\varepsilon Y$$

- Refined computation types:  $T_\varepsilon X$
- Erasure map  $U(\cdot)$  strips effect annotations off
- Well-formedness judgements  $\Pi \vdash \varepsilon \text{ ok}$ ,  $\Pi \vdash X \text{ ok}$  and  $\Pi \vdash \Theta \text{ ok}$  where  $\Pi \subseteq_{fin} Regs$
- Subtyping

$$\frac{}{X \leq X} \qquad \frac{X \leq Y \quad Y \leq Z}{X \leq Z} \qquad \frac{X \leq X' \quad Y \leq Y'}{X \times Y \leq X' \times Y'}$$

$$\frac{X' \leq X \quad T_\varepsilon Y \leq T_{\varepsilon'} Y'}{(X \rightarrow T_\varepsilon Y) \leq (X' \rightarrow T_{\varepsilon'} Y')}$$

$$\frac{\varepsilon \subseteq \varepsilon' \quad X \leq X'}{T_\varepsilon X \leq T_{\varepsilon'} X'}$$

# Refined type system: selected rules

- “Standard” rules, e.g.

$$\frac{\Pi; \Theta, x : X \vdash M : T_\varepsilon Y}{\Pi; \Theta \vdash \lambda x : U(X).M : X \rightarrow T_\varepsilon Y}$$

$$\frac{\Pi; \Theta \vdash M : T_\varepsilon X \quad \Pi; \Theta, x : X \vdash N : T_{\varepsilon'} Y}{\Pi; \Theta \vdash \text{let } x \leftarrow M \text{ in } N : T_{\varepsilon \cup \varepsilon'} Y}$$

- Rules for store instructions:

$$\frac{\Pi; \Theta \vdash V : \text{ref}_r}{\Pi; \Theta \vdash \text{read}(V) : T_{\{rd_r\}}(\text{int})}$$

$$\frac{\Pi; \Theta \vdash V_1 : \text{ref}_r \quad \Pi; \Theta \vdash V_2 : \text{int}}{\Pi; \Theta \vdash \text{write}(V_1, V_2) : T_{\{wr_r\}}(\text{unit})}$$

$$\frac{\Pi; \Theta \vdash V : \text{int} \quad r \in \Pi}{\Pi; \Theta \vdash \text{ref}(V) : T_{\{al_r\}}(\text{ref}_r)}$$

## Examples

$$V_{\text{sum}} := \lambda(x, y, z):\text{int}^3.\text{let } x \leftarrow \text{ref}(x + y) \text{ in } \text{read}(x) + z$$

$$: \text{int}^3 \rightarrow T_{\{al_r, rd_r\}} \text{int}$$

$$M_{\text{cnt}} := \text{let } x \leftarrow \text{ref}(0) \text{ in } (\lambda_- : \text{unit}.\text{read}(x), \lambda_- : \text{unit}.\text{write}(x, \text{read}(x) + 1))$$

$$: T_{\{al_r\}} ((\text{unit} \rightarrow T_{\{rd_r\}} \text{int}) \times (\text{unit} \rightarrow T_{\{wr_r, rd_r\}} \text{unit}))$$

$$M_{\text{mem}} := \text{let } x \leftarrow \text{ref}(0) \text{ in } \text{let } y \leftarrow \text{ref}(1) \text{ in}$$

$$\lambda a:\text{int}.\text{ if } a = \text{read}(x) \text{ then } \text{read}(y)$$

$$\text{ else } \text{write}(x, a); \text{write}(y, a + 1); \text{read}(y)$$

(Memoization of the successor function)

$$: T_{\{al_r\}} (\text{int} \rightarrow T_{\{wr_r, rd_r\}} \text{int})$$

$$M_{\text{buf}} := \text{let } x \leftarrow \text{ref}(0) \text{ in } \lambda a:\text{int}.\text{let } o \leftarrow \text{read}(x) \text{ in } \text{write}(x, a); o$$

(returns the argument of previous call, or zero)

$$: T_{\{al_r\}} (\text{int} \rightarrow T_{\{wr_r, rd_r\}} \text{int})$$

# “Forget” regions that are not externally relevant

## Masking rule

$$\frac{\Pi, r; \Theta \vdash M : T_\varepsilon X \quad \Pi \vdash X \text{ ok} \quad \Pi \vdash \Theta \text{ ok}}{\Pi; \Theta \vdash M : T_{\varepsilon \setminus \{rd_r, wr_r, al_r\}} X}$$

Applicability determined solely on the basis of the **typing**.

Allows e.g. to detect that

$$V_{\text{sum}} := \lambda(x, y, z):\text{int}^3.\text{let } x \leftarrow \text{ref}(x + y) \text{ in read}(x) + z$$

is pure, by assigning the type  $\text{int}^3 \rightarrow T_\emptyset \text{int}$  to it, instead of the previous type  $\text{int}^3 \rightarrow T_{\{al_r, rd_r\}} \text{int}$

Conservative approximation: purity of function  $M_{\text{mem}}$  is not discovered – indeed the **impure**  $M_{\text{buf}}$  has the same type!

# Parameters

Purpose: to serve as an index for the logical relation interpreting types (relation over varying “domain” due to memory allocation)

**Goal 1:** “representation independence”: invariance under permutation of locations

**Goal 2:** Distinction observable  $\leftrightarrow$  private locations

Assume special symbol  $\tau \notin \text{Regs}$  representing the silent region

## Definition (parameter)

A *parameter*  $\phi$  is a pair  $(\Pi, f)$  where

- $\Pi$  is a finite set of regions,
- function  $f$  assigns to each  $r \in \Pi \cup \{\tau\}$  a partial bijection  $f(r)$  such that distinct regions map to disjoint partial bijections.
- $f(\tau) = (L, L', \emptyset)$  for some  $L, L'$ .

# Store relations

## Definition (Store equivalence on location set)

For  $L \subseteq \mathbb{L}$  and  $s, s' \in \mathbb{S}$  define

$$s \sim_L s' \iff \text{dom}(s) \supseteq L \wedge \text{dom}(s') \supseteq L \wedge \forall l \in L. s.l = s'.l$$

## Definition (Store relation)

Let  $L, L' \subseteq \mathbb{L}$ . A *store relation* on  $L, L'$  is a nonempty relation  $R \subseteq \mathbb{S} \times \mathbb{S}$  such that

$$(s, s') \in R \wedge s \sim_L s_1 \wedge s' \sim_{L'} s'_1 \implies (s_1, s'_1) \in R.$$

We write  $\text{StRel}(L, L')$  for the set of all store relations on  $L, L'$ .

## Store relations respecting an effect

Let  $R$  be a store relation on  $\text{dom}(\phi)$ ,  $\text{dom}'(\phi)$ .  $R$  respects  $\varepsilon$  at  $\phi$  if it is preserved by all commands that exhibit only  $\varepsilon$ :

- $R$  respects  $\{rd_r\}$  at  $\phi$  if  $(s, s') \in R$  implies  $s.l = s'.l'$  for all  $(l, l') \in \phi(r)$ ;
- $R$  respects  $\{wr_r\}$  at  $\phi$  if for all  $(s, s') \in R$  and for all  $(l, l') \in \phi(r)$  and  $v \in \mathbb{Z}$ , we have  $(s[l \mapsto v], s'[l' \mapsto v]) \in R$ ;
- $R$  respects  $\{al_r\}$  always (tracked in the log. relation).

Set of store relations that respect  $\varepsilon$  at  $\phi$  (for  $\Pi \vdash \phi \text{ ok}$ ,  $\Pi \vdash \varepsilon \text{ ok}$ ):

$$\mathcal{R}_\varepsilon^\Pi(\phi) = \{R \in \text{StRel}(\text{dom}(\phi), \text{dom}'(\phi)) \mid \forall e \in \varepsilon, R \text{ resp. } e \text{ at } \phi\}.$$

### Definition

For  $s, s' \in \mathbb{S}$  define

- $s, s' \models \phi \iff \text{dom}(s) = \text{dom}(\phi) \wedge \text{dom}(s') = \text{dom}'(\phi)$
- $s \sim_\phi s' \iff \forall r \in \text{Regs}. \forall (l, l') \in \phi(r). s.l = s'.l'$

# Logical relation

Definition  $\llbracket X \rrbracket_{\phi}^{\Pi} \subseteq \llbracket U(X) \rrbracket \times \llbracket U(X) \rrbracket$

For  $\Pi \vdash X$  ok and  $\Pi \vdash \phi$  ok define

$$\llbracket X \rrbracket_{\phi}^{\Pi} \equiv \Delta_{\llbracket U(X) \rrbracket} \text{ when } X \in \{\text{int}, \text{bool}, \text{unit}\}$$

$$\llbracket \text{ref}_r \rrbracket_{\phi}^{\Pi} \equiv \phi(r)$$

$$\llbracket X \rightarrow T_{\varepsilon} Y \rrbracket_{\phi}^{\Pi} \equiv \{(f, f') \mid \forall \phi' \geq_{\Pi} \phi. \forall (x, x') \in \llbracket X \rrbracket_{\phi'}^{\Pi}. \\ (f(x), f'(x')) \in \llbracket T_{\varepsilon} Y \rrbracket_{\phi'}^{\Pi}\}$$

$$\llbracket T_{\varepsilon} X \rrbracket_{\phi}^{\Pi} \equiv \{(f, f') \mid \forall s, s', R. (s, s' \models \phi \wedge R \in \mathcal{R}_{\varepsilon}^{\Pi}(\phi) \wedge s R s') \\ \Rightarrow (\exists \psi \in \text{Par}(\text{als}(\varepsilon)). s_1, s'_1 \models \phi \otimes \psi \wedge \\ s_1 R s'_1 \wedge s_1 \sim_{\psi} s'_1 \wedge \\ (v, v') \in \llbracket X \rrbracket_{\phi \otimes \psi}^{\Pi}) \\ \text{where } (s_1, v) = f \text{ s and } (s'_1, v') = f' s'\}$$

where  $\psi \in \text{Par}(\text{als}(\varepsilon))$  means  $\text{regs}(\psi) \subseteq \{r \mid a/r \in \varepsilon\}$ .

Define  $\llbracket x_1 : X_1, \dots, x_n : X_n \rrbracket_{\phi}^{\Pi} \equiv \{(\gamma, \gamma') \mid \forall i. (\gamma(x_i), \gamma'(x_i)) \in \llbracket X_i \rrbracket_{\phi}^{\Pi}\}$

# Masking lemma and fundamental theorem

## Masking lemma

Suppose  $\Pi \vdash X \text{ ok}$  and  $\Pi, r \vdash \phi \text{ ok}$ . Then  $\llbracket X \rrbracket_{\phi}^{\Pi, r} = \llbracket X \rrbracket_{\phi-r}^{\Pi}$  and  $\llbracket T_{\varepsilon} X \rrbracket_{\phi}^{\Pi, r} = \llbracket T_{\varepsilon} X \rrbracket_{\phi-r}^{\Pi}$ .

where

$$(\phi-r)(r') = \phi(r') \text{ when } r' \neq r$$

$$(\phi-r)(\tau) = \phi(\tau) \otimes (\text{dom}(\phi(r)), \text{dom}'(\phi(r)), \emptyset)$$

## Fundamental theorem

Let  $\Pi \vdash \phi \text{ ok}$ ,  $\Pi; \Theta \vdash V : X$  and  $(\gamma, \gamma') \in \llbracket \Theta \rrbracket_{\phi}^{\Pi}$ . Then

$$(\llbracket U(\Theta) \vdash V : U(X) \rrbracket_{\gamma}, \llbracket U(\Theta) \vdash V : U(X) \rrbracket_{\gamma'}) \in \llbracket X \rrbracket_{\phi}^{\Pi}.$$

(and similarly for computations  $M$  with  $\Pi; \Theta \vdash M : T_{\varepsilon}(X)$ )

# Contextual equivalence

## Definition: contextual equivalence

- Let  $\Pi; \emptyset \vdash V_i : X$  for  $i = 1, 2$  be two closed values. Write  $\Pi; \emptyset \vdash V_1 \equiv V_2 : X$  if for all values  $\Pi; \emptyset \vdash V : X \rightarrow T_\varepsilon \text{bool}$  (“contexts”) with  $\varepsilon$  arbitrary it holds that when  $(s_1, v_1) = \llbracket V V_1 \rrbracket () \emptyset$  and  $(s_2, v_2) = \llbracket V V_2 \rrbracket () \emptyset$  then  $v_1 = v_2$ , where  $\emptyset$  is the initial, empty state.
- Extend to open values and computations by abstracting free variables. Notation:  $\Pi; \Theta \vdash M_1 \equiv M_2 : TX$

## Corollary: contextual equivalence

Suppose  $\Pi; \Theta \vdash M_i : T_\varepsilon X$  for  $i = 1, 2$ . Let

$$(\llbracket U(\Theta) \vdash M_1 : T(U(X)) \rrbracket \gamma, \llbracket U(\Theta) \vdash M_2 : T(U(X)) \rrbracket \gamma') \in \llbracket T_\varepsilon X \rrbracket_\phi^\Pi.$$

hold whenever  $\Pi \vdash \phi \text{ ok}$  and  $(\gamma, \gamma') \in \llbracket \Theta \rrbracket_\phi^\Pi$ . Then  $\Pi; \Theta \vdash M_1 \equiv M_2 : T_\varepsilon X$

## Application: program transformations (I)

Types determine applicability of transformation rules.

## Duplicated computations

Let  $\Pi; \Theta \vdash M : T_\varepsilon(X)$  and  $\text{rds}(\varepsilon) \cap \text{wrs}(\varepsilon) = \text{als}(\varepsilon) = \emptyset$ . Let

$$M_1 := \text{let } x \leftarrow M \text{ in val } (x, x)$$

$$M_2 := \text{let } x \leftarrow M \text{ in let } y \leftarrow M \text{ in val } (x, y).$$

Then  $\Pi; \Theta \vdash M_1 \equiv M_2 : T_\varepsilon(X \times X)$ .

## Commuting computations

Let  $\text{rds}(\varepsilon_1) \cap \text{wrs}(\varepsilon_2) = \text{rds}(\varepsilon_2) \cap \text{wrs}(\varepsilon_1) = \text{wrs}(\varepsilon_1) \cap \text{wrs}(\varepsilon_2) = \emptyset$  and  $\Pi; \Theta \vdash M_i : T_{\varepsilon_i}(X)$  for  $i \in \{1, 2\}$ . Let

$$N_1 := \text{let } y \leftarrow M_1 \text{ in let } x \leftarrow M_2 \text{ in val } (x, y)$$

$$N_2 := \text{let } x \leftarrow M_2 \text{ in let } y \leftarrow M_1 \text{ in val } (x, y)$$

and  $\varepsilon = \varepsilon_1 \cup \varepsilon_2$ . Then  $\Pi; \Theta \vdash N_1 \equiv N_2 : T_\varepsilon(X \times X)$

## Application: program transformations (II)

## Dead computation

Let  $\Pi; \Theta \vdash M : T_\varepsilon(\text{unit})$  and  $\text{wrs}(\varepsilon) = \emptyset$ . Then  
 $\Pi; \Theta \vdash M \equiv \text{val}() : T_\varepsilon(\text{unit})$ .

## Pure lambda hoist

Let  $\Pi; \Theta \vdash M : T_\emptyset Z$  and  $\Pi; \Theta, x:X, y:Z \vdash N : T_\varepsilon Y$ .  
Put

$$M_1 := \text{val}(\lambda x:UX. \text{let } y \leftarrow M \text{ in } N)$$

$$M_2 := \text{let } y \leftarrow M \text{ in val}(\lambda x:UX. N)$$

Then  $\Pi; \Theta \vdash M_1 \equiv M_2 : T_\emptyset (X \rightarrow T_\varepsilon Y)$

# Current work: general references

- Presented work limited to integer references
- Generalisation to references of arbitrary type desirable but non-trivial:
  - General recursion is expressible ( $\rightsquigarrow$  CPO's)
  - Interpretations of types are **admissible** relations
  - Closure of interpretations w.r.t. "zig-zag" property:  
 $xRy \& zRy \& zRw \Rightarrow xRw$
  - Homogeneous regions