

On stratified regions

Roberto M. Amadio

Université Paris Diderot (Paris 7)

Divergence in higher-order languages with side effects

```
# let l=ref(function (x: unit) ->x);;  
val l : (unit -> unit) ref = {contents = <fun>}
```

```
# l:=function (x:unit) -> (!l)x;;  
- : unit = ()
```

```
# !l ;;  
- : unit -> unit = <fun>
```

```
# (!l)();;  
Interrupted.
```

NB Similar phenomena arise with, *e.g.*, channels.

Can be used to define general recursion

```
# let rfact = ref(function (x:int) -> x);;
val rfact : (int -> int) ref = {contents = <fun>}

# rfact:= function(x:int)->if x=0 then 1 else x*(!rfact)(x-1) ;;
- : unit = ()

# let fact = !rfact ;;
val fact : int -> int = <fun>

# fact 3;;
- : int = 6
```

Stratification

Idea Associate a well-founded notion with rank to regions and make sure that what is written at a certain rank can only produce effects at lower ranks.

Approach Use *type and effect* systems

- Dynamic values are abstracted into *regions* r, r', \dots
- Effects e, e', \dots are finite sets of regions.
- We record the effect of a term.

NB This is due to Boudol (CONCUR 07).

Our purpose here is to show that this works for a *more abstract model* and with a *simpler interpretation*.

Syntax (types and contexts)

$A ::= \mathbf{1} \mid \text{Reg}_r A \mid (A \xrightarrow{e} A)$ (types)

$\Gamma = x_1 : A_1, \dots, x_n : A_n$ (context)

$R = r_1 : A_1, \dots, r_n : A_n$ (region context)

$R; \Gamma \vdash M : (A, e)$ (typing judgement)

Syntax (terms)

- Abstract dynamic values into regions (that is what the type system does anyway).
- The resulting language simulates references, channels, signals provided we take a *cumulative* semantics of side effects: what is written is added to what was written before.

$$M ::= x \mid r \mid * \mid \lambda x.M \mid MM \mid \text{get}(M) \mid \text{set}(M, M) \quad (\text{terms})$$
$$V ::= r \mid * \mid \lambda x.M \quad (\text{values})$$
$$S ::= (r \Leftarrow \{V_1, \dots, V_n, \dots\}) \mid S, S \quad (\text{stores})$$

A *program* is a multi-set of terms and stores.

Reduction rules

Denote with E a call-by-value *evaluation context*.

$$\frac{}{E[(\lambda x.M)V] \rightarrow E[[V/x]M]}$$

$$\frac{}{E[\text{get}(r)], (r \Leftarrow V) \rightarrow E[V], (r \Leftarrow V)}$$

$$\frac{}{E[\text{set}(r, V)] \rightarrow E[*], (r \Leftarrow V)}$$

$$\frac{P \rightarrow P'}{P, P'' \rightarrow P', P''}$$

Typing rules stratified or unstratified

$$\frac{R \vdash \Gamma \quad x : A \in \Gamma}{R; \Gamma \vdash x : (A, \emptyset)} \quad \frac{R \vdash \Gamma \quad r : A \in R}{R; \Gamma \vdash r : (\text{Reg}_r A, \emptyset)} \quad \frac{R \vdash \Gamma}{R; \Gamma \vdash * : (\mathbf{1}, \emptyset)}$$

$$\frac{R; \Gamma, x : A \vdash M : (B, e)}{R; \Gamma \vdash \lambda x. M : (A \xrightarrow{e} B, \emptyset)} \quad \frac{R; \Gamma \vdash M : (A \xrightarrow{e_2} B, e_1) \quad R; \Gamma \vdash N : (A, e_3)}{R; \Gamma \vdash MN : (B, e_1 \cup e_2 \cup e_3)}$$

$$\frac{R; \Gamma \vdash M : (\text{Reg}_r A, e)}{R; \Gamma \vdash \text{get}(M) : (A, e \cup \{r\})} \quad \frac{R; \Gamma \vdash M : (\text{Reg}_r A, e_1) \quad R; \Gamma \vdash N : (A, e_2)}{R; \Gamma \vdash \text{set}(M, N) : (\mathbf{1}, e_1 \cup e_2 \cup \{r\})}$$

To add: rules for subtyping and stores.

Region-contexts and types (stratified)

$$\frac{}{\emptyset \vdash} \quad \frac{R \vdash A \quad r \notin \text{dom}(R)}{R, r : A \vdash} \quad \frac{R \vdash}{R \vdash \mathbf{1}}$$

$$\frac{R \vdash \quad r : A \in R}{R \vdash \text{Reg}_r A} \quad \frac{R \vdash A \quad R \vdash B \quad e \subseteq \text{dom}(R)}{R \vdash A \xrightarrow{e} B}$$

$$\frac{R \vdash A \quad e \subseteq \text{dom}(R)}{R \vdash (A, e)} .$$

What types and what doesn't

Judgement	Unstratified	Stratified
$r : 1 \xrightarrow{\{r\}} 1 \vdash \text{Reg}_r(1 \xrightarrow{\{r\}} 1)$	yes	no
$r_1 : \text{Reg}_{r_2}(1 \xrightarrow{\{r_2\}} 1), r_2 : 1 \xrightarrow{\{r_1\}} 1 \vdash$	no	no
$r_1 : (1 \xrightarrow{\{r_2\}} 1), r_2 : (1 \xrightarrow{\{r_1\}} 1) \vdash$	yes	no
$r_1 : 1 \xrightarrow{\emptyset} 1, r_2 : 1 \xrightarrow{\{r_1\}} \text{Reg}_{r_1}(1 \xrightarrow{\emptyset} 1) \vdash$	yes	yes

Subject reduction

- Types and effects are preserved by reduction (because of subtyping).
- Effect delimitation: if $R; \vdash E[\text{get}(r)] : (A, e)$ then $r \in e$.
- If $R, R'; \vdash M : (A, e)$ and $R \vdash (A, e)$ then when running, applying, reading, ... M all the effects will be in $\text{dom}(R)$.
- However the typing of M may require R' . E.g.

$$M = (\lambda f.*)(\lambda x.\text{get}(r)x), \quad R = r : \mathbf{1} \xrightarrow{\emptyset} \mathbf{1}$$

Then

$$R; \vdash M : (\mathbf{1}, \emptyset) \quad \emptyset \vdash (\mathbf{1}, \emptyset) \quad \emptyset; \emptyset \not\vdash M : (\mathbf{1}, \emptyset)$$

Reducibility candidates for call-by-value (reminder)

Interpret types as sets of typable terms with certain properties:

$$\underline{\mathbf{1}} = \{M \mid \vdash M : \mathbf{1}, M \text{ terminates}\}$$

$$\underline{A \rightarrow B} = \{M \mid \vdash M : A \rightarrow B, \forall V \in \underline{A} (MV \in \underline{B})\}$$

The set \underline{A} is composed of (closed) terminating terms which are reduction closed, and expansion closed for *neutral* terms.

Then prove that:

$$x : A \vdash M : B \quad \text{implies} \quad \forall V \in \underline{A} \quad [V/x]M \in \underline{B}$$

Problems with the interpretation

A quote (Girard)

Conditions were originally found by trial and error.

First guess Region contexts are sets of stores and types are sets of terms.

Extension Write $R' \geq R$ if R' extends R .

The term $M = (\lambda f.*)(\lambda x.\text{get}(r)x)$ should belong to $\vdash (1, \emptyset)$.

Second guess

$$\begin{aligned}\underline{R \vdash} &= \{R' \vdash S \mid R' \geq R \dots\} \\ \underline{R \vdash (A, e)} &= \{R' \vdash M \mid R' \geq R \dots\}\end{aligned}$$

Implication Note that there is an extra-quantification:

$$\begin{aligned}\underline{R \vdash (A \xrightarrow{e} B, e')} &= \{R' \vdash M \mid \dots \\ &\quad \forall R'' \geq R' \ R'' \vdash V \in \underline{R \vdash (A, \emptyset)} \dots\}\end{aligned}$$

Invariant on the store

- Evaluation of a term depends and affects the store.
- Define (inductively) a *saturated store* for R (the largest store with domain $dom(R)$).
- A term running with a saturated store can simulate a term running with a smaller store.
- Moreover the saturated store cannot be affected by the term.

Region-context interpretation

Let $R = r_1 : A_1, \dots, r_n : A_n$ and

$R_{r_i} = r_1 : A_1, \dots, r_{i-1} : A_{i-1}$, for $i = 1, \dots, n$.

$$\underline{R} = \{ R' \vdash S \mid R' \geq R, \quad \text{dom}(S) = \text{dom}(R), \text{ and for } i = 1, \dots, n \\ S(r_i) = \{V \mid R' \vdash V \in \underline{R_{r_i}} \vdash (A_i, \emptyset)\} \}$$

If $R' \geq R$ then $\underline{R}(R')$ is defined as the store S such that $R' \vdash S \in \underline{R}$.

NB For $r \in \text{dom}(R)$ and $R = R_1, r : A, R_2$, $V \in \underline{R}(R')(r)$ means $R' \vdash V \in \underline{R_1} \vdash (A, \emptyset)$.

Type and effect interpretation

$$\begin{aligned}
 \underline{R \vdash (A, e)} = & \{ R' \vdash M \mid \\
 & (1) \quad R' \geq R, \quad R'; \emptyset \vdash M : (A, e), \\
 & (2) \quad \text{for all } R'' \geq R' \quad M, \underline{R}(R'') \in SN, \text{ and} \\
 & (3) \quad \text{for all } M', S', R'' \geq R' \quad (M, \underline{R}(R'')) \Downarrow (M', S') \\
 & \quad \text{implies } S' = \underline{R}(R'') \text{ and } \mathcal{C}(A, R, R'', M') \}
 \end{aligned}$$

where: $\mathcal{C}(A, R, R'', M') \equiv$

$$\begin{aligned}
 & (A = \mathbf{1} \quad \supset \quad M' = *) \wedge \\
 & (A = \text{Reg}_r B \quad \supset \quad M' = r) \wedge \\
 & (A = A_1 \xrightarrow{e'} A_2 \quad \supset \quad M' = \lambda x. N \quad \wedge \\
 & \quad \text{for all } R_1 \geq R'', R_1 \vdash V \in \underline{R \vdash (A_1, \emptyset)} \\
 & \quad \text{implies } R_1 \vdash M'V \in \underline{R \vdash (A_2, e')}).
 \end{aligned}$$

Basic properties

Weakening

$$\frac{R'' \geq R' \geq R, \quad R \vdash (A, e), \quad R' \vdash M \in \underline{R \vdash (A, e)}}{R'' \vdash M \in \underline{R \vdash (A, e)}}$$

Extension/Restriction

$$\frac{R'' \geq R' \geq R, \quad R \vdash (A, e)}{R'' \vdash M \in \underline{R \vdash (A, e)} \text{ iff } R'' \vdash M \in \underline{R' \vdash (A, e)}}$$

Subtyping

$$\frac{R \vdash (A, e) \leq (A', e')}{\underline{R \vdash (A, e)} \subseteq \underline{R \vdash (A', e')}}}$$

Strong normalisation

$$\frac{R' \vdash M \in \underline{R} \vdash (A, e), \quad R'' \geq R'}{M, \underline{R}(R'') \in SN}$$

Reduction closure

$$\frac{R' \vdash M \in \underline{R} \vdash (A, e), \quad R'' \geq R', \quad M, \underline{R}(R'') \rightarrow M', S'}{R'' \vdash M' \in \underline{R} \vdash (A, e), \quad S' = \underline{R}(R'')}$$

Non-emptiness

$$\frac{R \vdash A}{\exists V \forall R' \geq R \forall e \subseteq \text{dom}(R) \ R' \vdash V \in \underline{R \vdash (A, e)}}$$

Expansion closure

$$\frac{\begin{array}{l} R \vdash (A, e), \quad R' \geq R, \quad R'; \emptyset \vdash M : (A, e), \quad M \text{ neutral} \\ \forall R'' \geq R', M', S' \quad (M, \underline{R}(R'') \rightarrow M', S') \text{ implies} \\ (R'' \vdash M' \in \underline{R \vdash (A, e)}, \quad S' = \underline{R}(R'')) \end{array}}{R' \vdash M \in \underline{R \vdash (A, e)}}$$

Main theorem and the properties at work

$R; x : A \models M : (B, e)$ means $\forall R' \geq R$ and $R' \vdash V \in \underline{R \vdash (A, \emptyset)}$,
 $R' \vdash [V/x]M \in \underline{R \vdash (B, e)}$.

Main theorem If $R; \Gamma \vdash M : (B, e)$ then $R; \Gamma \models M : (B, e)$.

Hence the terms in the interpretation are exactly the typable ones.

Some cases of the proof

Suppose $R = R_1, r : A, R_2$.

write Suppose $R; \vdash \text{set}(r, V) : (\mathbf{1}, \{r\})$ follows from $R; \vdash V : (A, \emptyset)$.

By ind. hyp.: $R' \vdash V \in \underline{R} \vdash (A, \emptyset)$.

We need to show that $R' \vdash V \in \underline{R_1} \vdash (A, \emptyset)$ (store is not affected).

This follows from **restriction**.

read Suppose $R'; \vdash \text{get}(r) : (A, \{r\})$.

By **non-emptiness**, $\underline{R}(R')(r) \neq \emptyset$.

Thus $\text{get}(r), \underline{R}(R')$ reduces to $V, \underline{R}(R')$ for some value V such that $R' \vdash V \in \underline{R_1} \vdash (A, \emptyset)$.

We need to show that $R' \vdash V \in \underline{R} \vdash (A, \emptyset)$

This follows from **extension**.

Comparison with Boudol (Concur 07)

- We work with an abstract model (subsuming references, channels, and signals, and preemptive or cooperative scheduling).
- Direct interpretation (no need of a special order to define the interpretation).
- Simpler invariant on the store (the store is unchanged).
- We use candidates à la Girard rather than Stenlund-Tait (a matter of taste).
- Interpreting subtyping (not difficult but important in practice).

Some extensions

Parallel threads Immediate since threads running in parallel cannot affect the saturated store.

Thread generation Regard a multi-set of terms as a term and allow abstraction over such terms.

Timed/Synchronous extension Add an operator *else-next* to react to the termination of the computation.

$$\frac{R; \Gamma \vdash M : (A, e) \quad R; \Gamma \vdash N : (A, e')}{R; \Gamma \vdash M \triangleright N : (A, e)} .$$

NB We just record the effect of the term M in the first instant.

Derived fixed-point rule

Set

$$\text{fix}_r f.M = \lambda x.(\text{get}(\text{reg}_r(\lambda x.[\lambda x.\text{get}(r)x/f]M x))) x$$

Then the following rule is derived:

$$\frac{R; \Gamma, f : A \xrightarrow{e \cup \{r\}} B \vdash M : (A \xrightarrow{e} B, \emptyset) \quad r : A \xrightarrow{e} B \in R}{R; \Gamma \vdash \text{fix}_r f.M : (A \xrightarrow{e \cup \{r\}} B, \emptyset)}$$

Speculations

- It would be useful to add *inductive data types* and *iterators*.
- One could refine the typing to have a *confluent language* (similar to single-assignment references).
- For the confluent language, it seems possible to give *simple denotational models*.
- One could move from intuitionistic to an *affine-intuitionistic* framework (cf. work of Antoine Madet).
- One could further refine towards *elementary, light,...* logic.