Louis Mandel
INRIA, projet MOSCOVA

Réactivité en ReactiveML

Alidecs – 12/03/2007

Exemples

Premier exemple de programme non réactif :

```
let process p =
  loop () end
  ||
  pause; print_string "Hello"
```

Exemples

Premier exemple de programme non réactif :

```
let process p =
  loop () end
  ||
  pause; print_string "Hello"
Line 2, characters 2-13:
Warning: This expression may be an instantaneous loop.

let rec process q =
  run q
Line 2, characters 2-7:
Warning: This expression may produce an instantaneous recursion.
```

Objectif

Jusqu'à la version 1.05 de ReactiveML, il y avait seulement le test dynamique de la réactivité :

```
rmlc -n 42 -sampling 0.01 ...
```

But :

Définir une analyse statique simple permettant de détecter *la plupart* des boucles instantanées *sans trop* de faux avertissements.

Plan

1. Analyse d'instantanéité
2. Détection des récursions instantanées

Analyse d'instantanéité

```
ackerman 3 10 (* e1 *)
```

e1 est instantané.

```
print_int 1; pause; print_int 2 (* e2 *)
```

e2 est non instantané.

```
if x then pause else () (* e3 *)
```

e3 peut être instantané ou non.

Le langage

$$
\begin{aligned}
e \quad ::= \quad & x \mid c \mid (e, e) \mid \lambda x.e \mid e\ e \mid \texttt{rec}\ x\ \texttt{=}\ e \mid \texttt{process}\ e \\
& \mid e\texttt{;}e \mid \texttt{let}\ x\ \texttt{=}\ e\ \texttt{and}\ x\ \texttt{=}\ e\ \texttt{in}\ e \\
& \mid \texttt{signal}\ x\ \texttt{default}\ e\ \texttt{gather}\ e\ \texttt{in}\ e \\
& \mid \texttt{present}\ e\ \texttt{then}\ e\ \texttt{else}\ e \mid \texttt{emit}\ e\ e \mid \texttt{pause} \\
& \mid \texttt{run}\ e \mid \texttt{do}\ e\ \texttt{until}\ e(x)\ \texttt{->}\ e\ \texttt{done} \mid \texttt{do}\ e\ \texttt{when}\ e
\end{aligned}
$$

Exemple :

```
let rec process p = pause; run p
```

se traduit en : $\texttt{rec}\, p\, \texttt{=}\, \texttt{process}\ (\texttt{pause;run}\ p)$

On appelle p la variable de récursion.

Typage d'instantanéité

Les règles ont la forme :

$$e : k$$

où k ::= $-$ *instantané*

$\mid \pm$ *indéterminé*

$\mid +$ *non instantané*

L'ordre $<$ sur k est : $- < \pm < +$

$max(k_1, k_2)$ est définit par :

	$-$	$\pm$	$+$
$-$	$-$	$\pm$	$+$
$\pm$	$\pm$	$\pm$	$+$
$+$	$+$	$+$	$+$

$$c : - \qquad \frac{e : -}{\lambda x.e : -} \qquad \frac{e_1 : - \quad e_2 : -}{e_1\ e_2 : -}$$

$$\frac{e : -}{\texttt{rec}\, x\, \texttt{=}\, e : -} \qquad \frac{e : k}{\texttt{process}\ e : -} \qquad \frac{e_1 : k_1 \quad e_2 : k_2}{e_1\,\texttt{;}\,e_2 : max(k_1, k_2)}$$

$$\frac{e : - \qquad e_1 : k_1 \qquad e_2 : k_2}{\texttt{present}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : max(\pm, k_1)}$$

$$\texttt{pause} : + \qquad \frac{e : -}{\texttt{run}\ e : \pm}$$

Propriétés

Propriété 1

Si $e : -$ *et* $\emptyset \vdash e : \tau$ *et* $e/S \rightarrow^* e'/S'$ *et* e' *est une forme normale alors* e' *est une valeur*

Propriété 2

Si $e : +$ *et* $\emptyset \vdash e : \tau$ *et* $e/S \rightarrow^* e'/S'$ *et* e' *est une forme normale alors* e' *est une expression de fin d'instant mais* e' *n'est pas une valeur*

Limitations

```
let process dynamique s = loop present s then () else () end
```

Les boucles instantanées peuvent dépendre de propriétés dynamiques.

```
let process statique =
  loop
    signal s in present s then () else ()
  end
```

Le signal s ne peut pas être émis ($\Rightarrow$ analyse des potentiels).

```
let process p = pause
let process q = loop run p end
```

Le type du corps des processus est *oublié* par le typage.

Typage et instantanéité

Ajouter l'information d'instantanéité au type des processus : $\tau\,\texttt{process}^k$

Les règles ont la forme : $H \vdash e : \tau[k]$

Exemples de règles :

$$\frac{H \vdash e : \tau[k]}{H \vdash \texttt{process}\ e : (\tau\,\texttt{process}^k)[-]} \qquad \frac{H \vdash e : (\tau\,\texttt{process}^k)[-]}{H \vdash \texttt{run}\ e : \tau[k]}$$

$$\frac{H \vdash e : (\tau\,\texttt{process}^k)[k']}{H \vdash e : (\tau\,\texttt{process}^{\pm})[k']}$$

II Détection des boucles instantanées

Détection des boucles instantanées

Le cas `loop` :

```
loop e end
```

Si *e* n'est pas de type $+$ alors émettre un avertissement.

Le cas des récursions :

```
let rec process p = pause; run p
```

Tous les récursions doivent traverser un retard.

$\Rightarrow$ On va définir une analyse qui repose sur les noms des variables de récursion.

Exemples

Il faut faire attention aux alias.

```
let rec process p =
  let q = p in
  run q
```

Il faut faire attention aux appels de fonctions.

```
let rec process p =
  let q = f p in
  run q
```

Exemples

Il faut faire attention aux appels de fonctions et aux émissions.

```
let rec process p =
  signal s in
  emit s p;
  await immediate one s(x) in
  run x

let rec process q =
  let r = ref (process ()) in
  (fun x -> r := x) q;
  run !r
```

Exemples

Un appel récursif sous une abstraction n'est pas évalué.

```
let rec process p =
  let process q = run p in
  ()
```

Le système de type

Les règles ont la forme :

$$\Pi \vdash e : \pi$$

où $\Pi, \pi \quad ::= \quad \emptyset$

$$\mid x : n, \Pi \quad \text{avec } n \in \mathbb{N}$$

n est utiliser pour compter les niveaux d'abstraction.

$\Rightarrow$ la structure des types n'est pas gardée

Quelques opérations sur Π

$$\Pi(x) = \begin{cases} n & \text{si } \Pi = x' : n, \Pi' \text{ et } x = x' \\ \Pi'(x) & \text{si } \Pi = x' : n, \Pi' \text{ et } x \neq x' \\ +\infty & \text{si } \Pi = \emptyset \end{cases}$$

$\Pi = \Pi' + n$ ssi $\forall x.\Pi(x) = \Pi'(x) + n$ et $Dom(\Pi) = Dom(\Pi')$ et $n \in \mathbb{Z}$

$\Pi = \Pi'^{\downarrow}$ ssi $\Pi = \Pi' + (-1)$ $\quad$ $\Pi = \Pi'^{\uparrow}$ ssi $\Pi = \Pi' + 1$

$\Pi = \Pi_1 * \Pi_2$ ssi $\forall x.\Pi(x) = min(\Pi_1(x), \Pi_2(x))$ et $Dom(\Pi) = Dom(\Pi_1) \cup Dom(\Pi_2)$

$n = min(\Pi)$ ssi $\exists x \in Dom(\Pi).\Pi(x) = n$ et $\forall x'.min(\Pi(x), \Pi(x')) = n$

Quelques règles

$$\frac{x \notin Dom(\Pi)}{\Pi \vdash x : \emptyset} \qquad \frac{n = \Pi(x) \quad n > 0}{\Pi \vdash x : \{x : n\}} \qquad \frac{\Pi^{\uparrow} \vdash e : \pi}{\Pi \vdash \lambda x.e : \pi}$$

$$\frac{\Pi \vdash e_1 : \pi_1 \quad \Pi \vdash e_2 : \emptyset \quad \pi_1^{\downarrow} > 0}{\Pi \vdash e_1\, e_2 : \pi_1^{\downarrow}} \qquad \frac{x : 0, \Pi \vdash e : \pi}{\Pi \vdash \texttt{rec}\, x = e : \pi \backslash x}$$

$$\frac{\Pi \vdash e : \pi \quad \pi^{\downarrow} > 0}{\Pi \vdash \texttt{run}\ e : \pi^{\downarrow}}$$

Quelques règles

$$\frac{\Pi \vdash e_1 : \pi_1 \quad non\_instantaneous(e1) \quad \emptyset \vdash e_2 : \pi_2}{\Pi \vdash e_1 \,;\, e_2 : \pi_1}$$

$$\frac{\Pi \vdash e_1 : \pi_1 \quad not(non\_instantaneous(e1)) \quad \Pi \vdash e_2 : \pi_2}{\Pi \vdash e_1 \,;\, e_2 : \pi_1 * \pi_2}$$

Quelques règles

$$\frac{\Pi \vdash e_1 : \pi_1 \quad \Pi \vdash e_2 : \pi_2 \quad non\_instantaneous(e_1, e_2) \quad \emptyset \vdash e : \pi}{\Pi \vdash \texttt{let}\ x_1\ \texttt{=}\ e_1\ \texttt{and}\ x_2\ \texttt{=}\ e_2\ \texttt{in}\ e : \pi_1 * \pi_2}$$

$$\frac{\Pi \vdash e_1 : \pi_1 \quad \Pi \vdash e_2 : \pi_2 \quad not(non\_instantaneous(e_1, e_2)) \quad x_1 : min(\pi_1), x_2 : min(\pi_2), \Pi \vdash e : \pi}{\Pi \vdash \texttt{let}\ x_1\ \texttt{=}\ e_1\ \texttt{and}\ x_2\ \texttt{=}\ e_2\ \texttt{in}\ e : (\pi_1 * \pi_2 * \pi) \backslash x_1 \backslash x_2}$$

Propriété

Propriété 3

Si $\Pi \vdash e : \pi$ *et* $\emptyset \vdash e : \tau$ *et* $e : k$

alors $e/S \rightarrow^* e'/S'$ *termine et* e' *est une expression de fin d'instant.*

Problème :

Comment prouver le théoreme sans interdir les fonctions récursives ?

```
let rec fact x =
  if x > 1 then x * fact (x-1)
  else x
```

Limitations

On fait des restrictions fortes.

```
let rec process p =
  (fun x -> ()) p
Line 2, characters 2-17:
Warning: This expression may produce an instantaneous recursion.
```

Enfin, il peut y avoir des récurrences sans `rec`.

```
let f =
  let r = ref (process ()) in
  let process g = run !r in
  r := g;
  g
```

Conclusion

- Séparation entre l'analyse d'instantanéité et la détection des boucles.
- L'analyse est implantée dans la nouvelle version de ReactiveML :

`http://rml.inria.fr`

Perspectives

- Finir les preuves
- Collaboration avec Gérard Boudol
- Liens avec la causalité