Correction du Typage pour un Langage Synchrone Fonctionnel

A. Guatto A. Cohen L. Mandel M. Pouzet

GdT Langages, Types et Preuves
20 Octobre 2014

Il était une fois... les langages synchrones !

Comment...

- programmer des systèmes réactifs...
- dans un langage de haut niveau...
- et avec des garanties de sûreté ?

Lustre et ses descendants

- Langages purement fonctionnels avec des flots infinis
- Des analyses statiques dédiées pour vérifier **et** compiler
- Utilisés en production dans l'industrie

Types d'horloge

Le compilateur doit...

- borner l'occupation mémoire du programme
- assurer qu'il peut s'exécuter indéfiniment
- le transformer en une machine à état

Les types d'horloge...

- établissent une base de temps globale
- datent les éléments de chaque flot
- dirigent la génération de code

Cet exposé

Un langage minimaliste

- Une poignée de constructions
- Des types pauvres, des règles riches
- Fragment intéressant d'un langage plus complet

Objectifs

- Présenter intuitivement le langage
- Décrire son système de type
- Discuter sa métathéorie

Opérateurs point à point

```
f (x, y) = x * x + y
```

$$(\mathbf{int} :: (\mathbf{1})) \otimes (\mathbf{int} :: (\mathbf{1})) \multimap (\mathbf{int} :: (\mathbf{1}))$$

x	[0]	[1]	[2]	[3]	[4]	...
y	[0]	[3]	[1]	[2]	[4]	...
$f\ (x, y)$	[0]	[4]	[5]	[11]	[20]	...

$$\mathbf{int} :: \mathbf{1}(\mathbf{0}\ \mathbf{2}) \otimes \mathbf{int} :: \mathbf{1}(\mathbf{0}\ \mathbf{2}) \multimap \mathbf{int} :: \mathbf{1}(\mathbf{0}\ \mathbf{2})$$

x	[0]	[]	[1; 2]	[]	[3; 4]	...
y	[0]	[]	[3; 1]	[]	[2; 4]	...
$f\ (x, y)$	[0]	[]	[4; 5]	[]	[11; 20]	...

Typage des opérateurs point à point

$$\text{Op}\quad \frac{\Gamma_1 \vdash e_1 : \mathbf{int} :: ct \qquad \Gamma_2 \vdash e_2 : \mathbf{int} :: ct}{\Gamma_1 \otimes \Gamma_2 \vdash e_1 + e_2 : \mathbf{int} :: ct}$$

$$\begin{array}{rcl} t & ::= & dt :: ct \\ & | & t \otimes t \\ & | & t \multimap t \\ dt & ::= & \mathbf{bool} \mid \mathbf{int} \mid \dots \\ ct & ::= & p \\ & | & ct\ \mathbf{on}\ ct \\ p & ::= & c^*(c^+) \end{array}$$

Échantillonage

```
f (x, y) = x when (1 0) + y
```

$$\mathbf{int} :: (\mathbf{1}) \otimes \mathbf{int} :: (\mathbf{1}\ \mathbf{0}) \multimap \mathbf{int} :: (\mathbf{1}\ \mathbf{0})$$

x	[0]	[1]	[2]	[3]	[4]	[5]	...
y	[0]	[]	[1]	[]	[4]	[]	...
$f\ (x, y)$	[0]	[]	[3]	[]	[8]	[]	...

$$\mathbf{int} :: (\mathbf{2}) \otimes \mathbf{int} :: (\mathbf{1}) \multimap \mathbf{int} :: (\mathbf{1})$$

x	[0; 1]	[2; 3]	[4; 5]	...
y	[0]	[1]	[4]	...
$f\ (x, y)$	[0]	[3]	[8]	...

Typage de l'échantillonage

$$\text{When}\quad \frac{\Gamma \vdash e : dt :: ct \qquad \vdash p : \mathbf{bool} :: ct}{\Gamma \vdash e\ \texttt{when}\ p : dt :: ct\ \mathbf{on}\ p}$$

$$(n.w)\ on\ (m_1 \dots m_n.w') = (\sum_{1 \le i \le n} m_i).(w\ on\ w')$$

$$\begin{array}{rll} \texttt{f} : & \mathbf{int} :: (\mathbf{1}) \otimes \mathbf{int} :: (\mathbf{1})\ \mathbf{on}\ (\mathbf{1}\ \mathbf{0}) & \multimap \mathbf{int} :: (\mathbf{1})\ \mathbf{on}\ (\mathbf{1}\ \mathbf{0}) \\ \equiv & \mathbf{int} :: (\mathbf{1}) \otimes \mathbf{int} :: (\mathbf{1}\ \mathbf{0}) & \multimap \mathbf{int} :: (\mathbf{1}\ \mathbf{0}) \end{array}$$

$$\begin{array}{rll} \texttt{f} : & \mathbf{int} :: (\mathbf{2}) \otimes \mathbf{int} :: (\mathbf{2})\ \mathbf{on}\ (\mathbf{1}\ \mathbf{0}) & \multimap \mathbf{int} :: (\mathbf{2})\ \mathbf{on}\ (\mathbf{1}\ \mathbf{0}) \\ \equiv & \mathbf{int} :: (\mathbf{2}) \otimes \mathbf{int} :: (\mathbf{1}) & \multimap \mathbf{int} :: (\mathbf{1}) \end{array}$$

Fusion ; sous-typage

```
s x = merge (1 0) x x
```

int :: **(1)** $\multimap$ **int** :: **(2)**

x	[0]	[1]	[2]	...
$s\ x$	[0; 0]	[1; 1]	[2; 2]	...

int :: **(1 0)** $\multimap$ **int** :: **(1)**

x	[0]	[]	[1]	[]	[2]	[]	...
$s\ x$	[0]	[0]	[1]	[1]	[2]	[2]	...

Typage de la fusion et sous-typage

Merge

$$\frac{\vdash p : \mathbf{bool} :: ct \qquad \Gamma_1 \vdash e_1 : dt :: ct\ \mathbf{on}\ p \qquad \Gamma_2 \vdash e_2 : dt :: ct\ \mathbf{on}\ (not\ p)}{\Gamma_1 \otimes \Gamma_2 \vdash \mathtt{merge}\ p\ e_1\ e_2 : dt :: ct}$$

Sub

$$\frac{\Gamma \vdash e : t \qquad \vdash t <:_k t'}{\Gamma \vdash e : t'}$$

$$w <:_k w' \quad = \quad \exists n \in \mathbb{N}, \forall i \in \mathbb{N}, 0 \leq \mathcal{O}_w(i) - \mathcal{O}_{w'}(i+k) \leq n$$
$$\text{avec} \quad \mathcal{O}_w(0) \quad = \quad 0$$
$$\mathcal{O}_{n.w}(1+i) \quad = \quad n + \mathcal{O}_w(i)$$

Typage de la fusion et sous-typage ; exemple

Sans les types de données, e.g. **int** :: **(1)** abstrait en **(1)** :

$$\frac{\mathtt{x} : \mathbf{(1)} \vdash \mathtt{x} : \mathbf{(2)}\ \mathbf{on}\ \mathbf{(1\ 0)} \qquad \mathtt{x} : \mathbf{(1)} \vdash \mathtt{x} : \mathbf{(2)}\ \mathbf{on}\ \mathbf{(0\ 1)}}{\mathtt{x} : \mathbf{(1)} \vdash \mathtt{merge}\ (1\ 0)\ \mathtt{x}\ \mathtt{x} : \mathbf{(2)}}\ \text{Merge}$$

vs.

$$\frac{\dfrac{\mathtt{x} : \mathbf{(1\ 0)} \vdash \mathtt{x} : \mathbf{(1\ 0)}}{\mathtt{x} : \mathbf{(1\ 0)} \vdash \mathtt{x} : \mathbf{(1\ 0)}} \qquad \dfrac{\mathtt{x} : \mathbf{(1\ 0)} \vdash \mathtt{x} : \mathbf{(1\ 0)} \qquad \vdash \mathbf{(1\ 0)} <:_1 \mathbf{(0\ 1)}}{\mathtt{x} : \mathbf{(1\ 0)} \vdash \mathtt{x} : \mathbf{(0\ 1)}}\ \text{Sub}}{\mathtt{x} : \mathbf{(1\ 0)} \vdash \mathtt{merge}\ (1\ 0)\ \mathtt{x}\ \mathtt{x} : \mathbf{(1)}}\ \text{Merge}$$

Points fixes

```
n x = merge 1(0) 0 (1 + x)
```

int :: **0(1)** $\multimap$ **int** :: **(1)**

x	[]	[4]	[2]	[11]	[3]	[3]	...
$n\ x$	[0]	[5]	[4]	[12]	[4]	[4]	...

```
nat = fix n
```

int :: **(1)**

nat	[0]	[1]	[2]	[3]	[4]	[5]	...

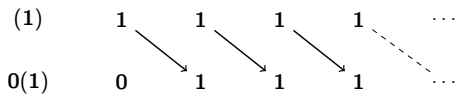
Typage des points fixes

Fix

$$\frac{\Gamma \vdash e : t \multimap t' \qquad \vdash t' <:_1 t \qquad rate(t') > 0}{\Gamma \vdash \mathtt{fix}\ e : t'}$$

Dans l'exemple précédent :

$$\mathbf{(1)} <:_1 \mathbf{0(1)}$$



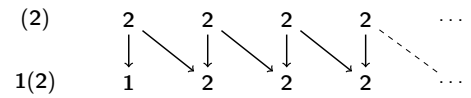
Productivité et modularité (1/2)

```
n' x = merge 1(0) 0 (1 + x)
```

int :: **1(2)** $\multimap$ **int** :: **(2)**

x	[4]	[2; 11]	[3; 3]	...
$n'\ x$	[0; 5]	[3; 12]	[4; 4]	...

```
nat' = fix n'
```
mal typé : $\neg\ \mathbf{(2)} <:_1 \mathbf{1(2)}$



Productivité et modularité (2/2)

```
sum x = x when (1 0) + x when (0 1)
```

int :: **(2)** $\multimap$ **int** :: **(1)**

Peut-on accepter le programme suivant avec `n` : **int** :: **0(1)** ?

```
sum (fix (\n.merge 1(0) 0 (1 + n)))
```

Oui : il faut accélérer le calcul de $\lambda \mathtt{n}.e$!

Temps local

By

$$\frac{\Gamma' \vdash e : t' \qquad \vdash \Gamma' \uparrow_p \Gamma \qquad \vdash t' \uparrow_p t}{\Gamma \vdash e : t}$$

UpStream

$$\frac{}{\vdash dt :: ct \uparrow_p dt :: p\ \mathbf{on}\ ct}$$

UpPair

$$\frac{\vdash t_1 \uparrow_p t_1' \qquad \vdash t_2 \uparrow_p t_2'}{\vdash t_1 \otimes t_2 \uparrow_p t_1' \otimes t_2'}$$

(N.B. : restreint au premier ordre par construction.)

Temps local, exemple 1/2

$$\dfrac{\dfrac{\dots}{\Box \vdash e_1 : (\mathbf{2}) \multimap (\mathbf{1})} \qquad \dfrac{\dfrac{\dots}{\dots \quad \vdash (\mathbf{1}) <:_1 \mathbf{0}(\mathbf{1})}}{\Box \vdash \mathtt{fix}\ \lambda \mathtt{n}.e_2 : (\mathbf{1}) \quad \vdash \Box \uparrow_{(\mathbf{2})} \Box} \quad \vdash (\mathbf{1}) \uparrow_{(\mathbf{2})} (\mathbf{2})}{\Box \vdash e_1\ (\mathtt{fix}\ \lambda \mathtt{n}.e_2) : (\mathbf{1})} \qquad \Box \vdash \mathtt{fix}\ \lambda \mathtt{n}.e_2 : (\mathbf{2})$$

avec

$$\begin{array}{rcl} e_1 & = & \lambda \mathtt{x}.(\mathtt{x\ when}\ (\mathbf{1}\ \mathbf{0}) + \mathtt{x\ when}\ (\mathbf{0}\ \mathbf{1})) \\ e_2 & = & \mathtt{merge}\ \mathbf{0}(\mathbf{1})\ 0\ (1 + \mathtt{n}) \end{array}$$

Temps local, exemple 2/2

```
csum x = fix (\y.x + merge 1(0) 0 y)
```

$$\mathbf{int} :: (\mathbf{2}) \multimap \mathbf{int} :: (\mathbf{2})$$

$$\dfrac{\dfrac{\dfrac{\dots}{x : (\mathbf{1}) \vdash \lambda y.e : \mathbf{0}(\mathbf{1}) \multimap (\mathbf{1})} \quad \vdash (\mathbf{1}) <:_1 \mathbf{0}(\mathbf{1})}{x : (\mathbf{1}) \vdash \mathtt{fix}(\lambda y.e) : (\mathbf{1})} \quad \vdash (\mathbf{2}) \uparrow_{(\mathbf{2})} (\mathbf{1})}{x : (\mathbf{2}) \vdash \mathtt{fix}(\lambda y.e) : (\mathbf{2})}}{\Box \vdash \lambda x.\mathtt{fix}(\lambda y.e) : (\mathbf{2}) \multimap (\mathbf{2})}$$

avec

$$e \quad = \quad \mathtt{x} + \mathtt{merge}\ \mathbf{1}(\mathbf{0})\ 0\ \mathtt{y}$$

Le langage, formellement

- Langage fonctionnel avec ordre supérieur linéaire
  - Fonctions appliquées exactement une fois
  - Flots duplicables et effaçables
- Sémantiques. . .
  - de Kahn : uniquement des flots
  - synchrone : du temps, des horloges
- Correction : les types sont des approximations correctes
  - Preuve par réalisabilité

Syntaxe

$$\begin{array}{rcl@{\qquad}rcl} e & ::= & x & t & ::= & dt :: ct \\ & | & \lambda x.e & & | & t \otimes t \\ & | & e\ e & & | & t \multimap t \\ & | & (e, e) & dt & ::= & \mathbf{bool} \mid \mathbf{int} \mid \dots \\ & | & \mathtt{let}\ (x, x) = e\ \mathtt{in}\ e & ct & ::= & p \\ & | & \mathtt{fix}\ e & & | & ct\ \mathbf{on}\ ct \\ & | & c & & & \\ & | & op\ e & & & \\ & | & \mathtt{merge}\ p\ e\ e & & & \\ & | & e\ \mathtt{when}\ p & \Gamma & ::= & \Box \\ p & ::= & c^*(c^+) & & | & \Gamma, x : t \end{array}$$

Typage (1/4)

Var
$$\dfrac{\vdash \Gamma\ \text{erasable}}{\Gamma, x : t \vdash x : t}$$

Weaken
$$\dfrac{\Gamma \vdash e : t \qquad \vdash t'\ \text{erasable} \qquad x \notin FV(e)}{\Gamma, x : t' \vdash e : t}$$

Lambda
$$\dfrac{\Gamma, x : t \vdash e : t'}{\Gamma \vdash \lambda x.e : t \multimap t'}$$

App
$$\dfrac{\Gamma \vdash \Gamma_1 \otimes \Gamma_2 \qquad \Gamma_1 \vdash e : t \multimap t' \qquad \Gamma_2 \vdash e' : t}{\Gamma \vdash e\ e' : t'}$$

Typage (2/4)

Pair
$$\dfrac{\Gamma \vdash \Gamma_1 \otimes \Gamma_2 \qquad \Gamma_1 \vdash e_1 : t_1 \qquad \Gamma_2 \vdash e_2 : t_2}{\Gamma \vdash (e_1, e_2) : t_1 \otimes t_2}$$

LetPair
$$\dfrac{\Gamma \vdash \Gamma_1 \otimes \Gamma_2 \qquad \Gamma_1 \vdash e : t_1 \otimes t_2 \qquad \Gamma_2, x : t_1, y : t_2 \vdash e' : t}{\Gamma \vdash \mathtt{let}\ (x, y) = e\ \mathtt{in}\ e' : t}$$

Fix
$$\dfrac{\Gamma \vdash e : t \multimap t' \qquad \vdash t' <:_1 t \qquad \mathit{rate}(t') > 0}{\Gamma \vdash \mathtt{fix}\ e : t'}$$

Typage (3/4)

ConstInt
$$\dfrac{}{\Box \vdash n : \mathbf{int} :: ct}$$

$\dots$

Op
$$\dfrac{\Gamma \vdash e : \mathbf{int} :: ct \otimes \mathbf{int} :: ct}{\Gamma \vdash op\ e : \mathbf{int} :: ct}$$

When
$$\dfrac{\Gamma \vdash e : dt :: ct \qquad \vdash p : \mathbf{bool} :: ct}{\Gamma \vdash e\ \mathtt{when}\ p : dt :: ct\ \mathbf{on}\ p}$$

Merge
$$\dfrac{\vdash p : \mathbf{bool} :: ct \qquad \Gamma_1 \vdash e_1 : dt :: ct\ \mathbf{on}\ p \qquad \Gamma_2 \vdash e_2 : dt :: ct\ \mathbf{on}\ (not\ p)}{\Gamma_1 \otimes \Gamma_2 \vdash \mathtt{merge}\ p\ e_1\ e_2 : dt :: ct}$$

Typage (4/4)

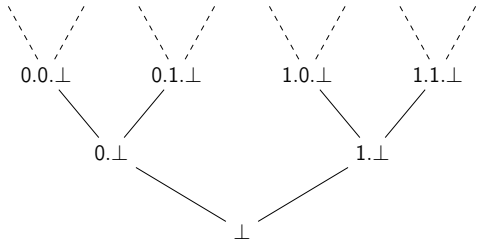
Sub
$$\dfrac{\Gamma \vdash e : t \qquad \vdash t <:_k t'}{\Gamma \vdash e : t'}$$

By
$$\dfrac{\Gamma' \vdash e : t' \qquad \vdash \Gamma' \uparrow_p \Gamma \qquad \vdash t' \uparrow_p t}{\Gamma \vdash e : t}$$

Sémantique : domaines (1/2)

$$Stream(A) \cong A \otimes Stream(A)_\perp$$

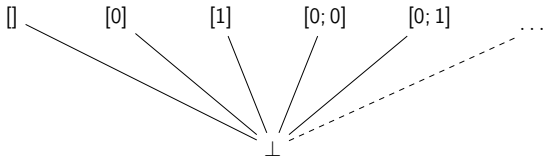
Par exemple, $Stream(\mathbb{B}_\perp)$:



Sémantique : domaines (2/2)

$$List(A) \cong 1 \oplus A \otimes List(A)$$

Par exemple, $List(\mathbb{B}_\perp)$:



Sémantique : interprétation des types et contextes

$$\begin{array}{lcl}
\mathcal{K}[\![dt :: ct]\!] & = & Stream([\![dt]\!]) \\
\mathcal{K}[\![t_1 \otimes t_2]\!] & = & \mathcal{K}[\![t_1]\!] \times \mathcal{K}[\![t_2]\!] \\
\mathcal{K}[\![t_1 \multimap t_2]\!] & = & \mathcal{K}[\![t_1]\!] \Rightarrow_c \mathcal{K}[\![t_2]\!]
\end{array}$$

$$\begin{array}{lcl}
\mathcal{K}[\![\square]\!] & = & 1 \\
\mathcal{K}[\![\Gamma, x : t]\!] & = & \mathcal{K}[\![\Gamma]\!] \times \mathcal{K}[\![t]\!]
\end{array}$$

$$\begin{array}{lcl}
\mathcal{S}[\![dt :: ct]\!] & = & Stream(List([\![dt]\!])) \\
\mathcal{S}[\![t_1 \otimes t_2]\!] & = & \mathcal{S}[\![t_1]\!] \times \mathcal{S}[\![t_2]\!] \\
\mathcal{S}[\![t_1 \multimap t_2]\!] & = & \mathcal{S}[\![t_1]\!] \Rightarrow_c \mathcal{S}[\![t_2]\!]
\end{array}$$

$$\begin{array}{lcl}
\mathcal{S}[\![\square]\!] & = & 1 \\
\mathcal{S}[\![\Gamma, x : t]\!] & = & \mathcal{S}[\![\Gamma]\!] \times \mathcal{S}[\![t]\!]
\end{array}$$

Sémantique : interprétation des programmes

Par induction sur les dérivations de typage :

$$\begin{array}{lcl}
\mathcal{K}[\![\Gamma \vdash e : t]\!] & : & \mathcal{K}[\![\Gamma]\!] \to_c \mathcal{K}[\![t]\!] \\
\mathcal{S}[\![\Gamma \vdash e : t]\!] & : & \mathcal{S}[\![\Gamma]\!] \to_c \mathcal{S}[\![t]\!]
\end{array}$$

La sémantique de Kahn est cependant *cohérente* (Reynolds).

Par ailleurs, l'adaptabilité joue un rôle calculatoire dans la sémantique synchrone :

$$\mathcal{S}[\![\vdash t <:_k t']\!] \ : \ \mathcal{S}[\![t]\!] \to_c \mathcal{S}[\![t']\!]$$

Sémantique : interprétation des constantes

$$\begin{array}{lcl}
\mathcal{K}[\![\square \vdash c : \mathbf{int} :: ct]\!] & = & repeat\ c \\
\mathcal{S}[\![\square \vdash c : \mathbf{int} :: ct]\!] & = & pack_{[\![ct]\!]}\ (repeat\ c)
\end{array}$$

avec

$$\begin{array}{lcl}
pack_{(n.w)}\ xs & = & (take\ n\ xs).(pack_w\ (drop\ n\ xs)) \\
\\
take\ 0\ xs & = & [] \\
take\ (1+n)\ (x.xs) & = & x; (take\ n\ xs) \\
\\
drop\ 0\ xs & = & xs \\
drop\ (1+n)\ (x.xs) & = & drop\ n\ xs
\end{array}$$

Sémantique : interprétation de l'échantillonage

$$\begin{array}{lcl}
\mathcal{K}[\![\Gamma \vdash e \texttt{ when } p : dt :: ct \textbf{ on } p]\!] & = & \lambda\gamma.(xs\ when\ cs) \\
 & & \quad xs = \mathcal{K}[\![\Gamma \vdash e : dt :: ct]\!]\ \gamma \\
 & & \quad cs = \mathcal{K}[\![\vdash p : dt :: ct]\!] \\
\mathcal{S}[\![\Gamma \vdash e \texttt{ when } p : dt :: ct \textbf{ on } p]\!] & = & \lambda\gamma.(map_2\ when^{\mathcal{S}}\ xs\ cs) \\
 & & \quad xs = \mathcal{S}[\![\Gamma \vdash e : dt :: ct]\!]\ \gamma \\
 & & \quad cs = \mathcal{S}[\![\vdash p : dt :: ct]\!]
\end{array}$$

avec

$$\begin{array}{lcl}
(x.xs)\ when\ (1.cs) & = & x.(xs\ when\ cs) \\
(x.xs)\ when\ (0.cs) & = & xs\ when\ cs \\
\\
xl\ when^{\mathcal{S}}\ cl & = & take\ |cl|_1\ ((unroll\ xl)\ when\ (unroll\ cl)) \\
\\
unroll\ [] & = & unroll\ [] \\
unroll\ (x; xl) & = & x.(unroll\ xl))
\end{array}$$

Sémantique : interprétation du sous-typage

$$\begin{array}{lcl}
\mathcal{K}[\![\Gamma \vdash e : t']\!] & = & \mathcal{K}[\![\Gamma \vdash e : t]\!] \\
\mathcal{S}[\![\Gamma \vdash e : t']\!] & = & \mathcal{S}[\![\vdash t <:_k t']\!] \circ \mathcal{S}[\![\Gamma \vdash e : t]\!]
\end{array}$$

$$\begin{array}{lcl}
\mathcal{S}[\![\vdash dt :: ct <:_k dt :: ct']\!] & = & pack_{[\![ct']\!]} \circ unpack \\
\mathcal{S}[\![\vdash t_1 \otimes t_2 <:_k t_1' \otimes t_2']\!] & = & \lambda(xs_1, xs_2).(ys_1, ys_2) \\
 & & \quad ys_1 = \mathcal{S}[\![\vdash t_1 <:_k t_1']\!]\ xs_1 \\
 & & \quad ys_2 = \mathcal{S}[\![\vdash t_2 <:_k t_2']\!]\ xs_2
\end{array}$$

avec

$$\begin{array}{lcl}
unpack\ ([].xs) & = & unpack\ xs \\
unpack\ ((x; xl).xs) & = & x.(unpack\ (xl.xs))
\end{array}$$

Sémantique : interprétation des points fixes

$$\begin{array}{lcl}
\mathcal{K}[\![\Gamma \vdash \texttt{fix}\ e : t']\!] & = & \lambda\gamma.fix\ (\mathcal{K}[\![\Gamma \vdash e : t]\!]\ \gamma) \\
\mathcal{S}[\![\Gamma \vdash \texttt{fix}\ e : t']\!] & = & \lambda\gamma.fix\ (\mathcal{S}[\![\Gamma \vdash e : t]\!]\ \gamma \circ \mathcal{S}[\![\vdash t' <:_k t]\!])
\end{array}$$

Correction

- Le système de type doit permettre la compilation séparée vers des machines à état fini.
- On se contente dans cet exposé d'une première étape.

Correction pour la sémantique synchrone

Les types approximent correctement le comportement calculatoire de la sémantique synchrone, i.e. pour toute expression close e et types dt et ct, on a

$$clock\ \mathcal{S}[\![\Box \vdash e : dt :: ct]\!] = [\![ct]\!]$$

avec

$$\begin{array}{lcl} clock & : & Stream(List(A)) \to_c Stream(\mathbb{N}_\perp) \\ clock & = & map_1\ length \end{array}$$

Correction : preuve (1/2)

On procède par *réalisabilité*.

- On définit l'ensemble des *réaliseurs* d'un type t :

$$\begin{array}{lcl} \mathcal{W}_t & \subseteq & \mathcal{S}[\![t]\!] \\ \mathcal{W}_{dt :: ct} & = & \{xs \in \mathcal{S}[\![dt :: ct]\!] \mid clock\ xs = [\![ct]\!]\} \\ \mathcal{W}_{t_1 \otimes t_2} & = & \mathcal{W}_{t_1} \times \mathcal{W}_{t_2} \\ \mathcal{W}_{t \multimap t'} & = & \{f \in \mathcal{S}[\![t \multimap t']\!] \mid \forall x \in \mathcal{W}_t, (f\ x) \in \mathcal{W}_{t'}\} \\ \mathcal{W}_\Gamma & \subseteq & \mathcal{S}[\![\Gamma]\!] \\ & \dots & \end{array}$$

- Le résultat précédent devient un corollaire du *lemme d'adéquation* : pour tout Γ, e et t, on a

$$\forall \gamma \in \mathcal{W}_\Gamma, (\mathcal{S}[\![\Gamma \vdash e : t]\!]\ \gamma) \in \mathcal{W}_t$$

- Malheureusement cela ne fonctionne pas tout à fait !

Correction : preuve (2/2)

- La preuve achoppe sur le cas des points fixes : on a besoin d'information sur les flots finis.
- On raffine les réaliseurs comme suit :

$$\begin{array}{lcl} \mathcal{W}_t^{n \in \mathbb{N}} & \subseteq & \mathcal{S}[\![t]\!] \\ \mathcal{W}_{dt :: ct}^n & = & \{xs \in \mathcal{S}[\![dt :: ct]\!] \mid clock\ xs =_n [\![ct]\!]\} \\ \mathcal{W}_{t_1 \otimes t_2}^n & = & \mathcal{W}_{t_1}^n \times \mathcal{W}_{t_2}^n \\ \mathcal{W}_{t \multimap t'}^n & = & \{f \in \mathcal{S}[\![t \multimap t']\!] \mid \forall x \in \mathcal{W}_t^n, (f\ x) \in \mathcal{W}_{t'}^n\} \\ \mathcal{W}_\Gamma^{n \in \mathbb{N}} & \subseteq & \mathcal{S}[\![\Gamma]\!] \\ & \dots & \end{array}$$

- On reformule le lemme d'adéquation :

$$\forall n \in \mathbb{N}, \forall \gamma \in \mathcal{W}_\Gamma^n, (\mathcal{S}[\![\Gamma \vdash e : t]\!]\ \gamma) \in \mathcal{W}_t^n$$

- Un outil essentiel pour les points fixes :

$$\forall t, t', \forall k, n \in \mathbb{N}, \forall xs \in \mathcal{W}_t^n, (\mathcal{S}[\![\vdash t <:_k t']\!]\ xs) \in \mathcal{W}_t^{n+k}$$

Reste à faire...

- Démontrer que les deux sémantiques coïncident, i.e. que pour toute expression close e et types dt et ct :

$$\mathcal{S}[\![\Box \vdash e : dt :: ct]\!] = pack_{[\![ct]\!]}\ \mathcal{K}[\![\Box \vdash e : dt :: ct]\!]$$

- Comprendre l'interaction entre l'accélération et l'ordre supérieur ; la règle "UpArrow" naïve est incorrecte.
- Étudier des types d'horloge plus riches

Travaux connexes

Construit sur...

- Lustre (Caspi, Halbwachs...)
- Lucid Synchrone (Pouzet)
- Lucy-n (Mandel, Plateau, Pouzet)

Points distinctifs

- Horloges entières
- Temps local
- Productivité basée sur les types
- (Ordre supérieur linéaire)

Conclusion

Un langage...

- Minimaliste
- Avec du temps flexible
- Doté de bonnes propriétés

Autres aspects

- Inférence de type
- Compilation vers machines à état

Annexes

Sémantique des types d'horloge et conditions

$$\begin{array}{lcl} [\![u(v)]\!] & = & upw\ u\ v \\ & & \text{avec}\ \ upw\ []\ v = upw\ v\ v \\ & & \quad\ upw\ (x; u)\ v = x.(upw\ u\ v) \\ [\![ct_1\ \textbf{on}\ ct_2]\!] & = & [\![ct_1]\!]\ on\ [\![ct_2]\!] \end{array}$$

$$\begin{array}{lcl} \mathcal{K}[\![\vdash p : dt :: ct]\!] & = & [\![p]\!] \\ \mathcal{S}[\![\vdash p : dt :: ct]\!] & = & pack_{[\![ct]\!]}\ [\![p]\!] \end{array}$$

Jugement d'adaptabilité

AdaptStream

$$\frac{nf(ct) <:_k nf(ct')}{\vdash dt :: ct <:_k dt :: ct'}$$

AdaptProd

$$\frac{\vdash t_1 <:_k t_1' \qquad \vdash t_2 <:_k t_2'}{\vdash t_1 \otimes t_2 <:_k t_1' \otimes t_2'}$$

Jugement de séparation

SepEmpty

$$\frac{}{\square \vdash \square \otimes \square}$$

SepContract

$$\frac{\Gamma \vdash \Gamma_1 \otimes \Gamma_2 \qquad \vdash t \ \mathit{duplicable}}{\Gamma, x : t \vdash \Gamma_1, x : t \otimes \Gamma_2, x : t}$$

SepLeft

$$\frac{\Gamma \vdash \Gamma_1 \otimes \Gamma_2 \qquad x \notin \mathit{Defs}(\Gamma_2)}{\Gamma, x : t \vdash \Gamma_1, x : t \otimes \Gamma_2}$$

SepRight

$$\frac{\Gamma \vdash \Gamma_1 \otimes \Gamma_2 \qquad x \notin \mathit{Defs}(\Gamma_1)}{\Gamma, x : t \vdash \Gamma_1 \otimes \Gamma_2, x : t}$$