Mini-ML

Mini language fonctionnel, pur:

$$\begin{array}{lrcl} \text{(\textit{types})} & \tau & ::= & \texttt{Int} \mid \texttt{Bool} \mid \tau_1 \to \tau_2 \\ \text{(\textit{exps})} & e & ::= & c \mid x \mid \lambda x \to e \mid e_1\ e_2 \mid e : \tau \\ & & & \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \mid \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \end{array}$$

Ajout d'effets de bord:

$$\begin{array}{lrcl} \text{(\textit{types})} & \tau & ::= & \ldots \mid \texttt{ref}\ \tau \mid \texttt{Unit} \\ \text{(\textit{expressions})} & e & ::= & \ldots \mid \texttt{ref}\ e \mid !e \mid e_1 := e_2 \mid e_1; e_2 \end{array}$$

Typage bidirectionnel

Au lieu de

$$\Gamma \vdash e : \tau$$

Le typage bidirectionnel utilise 2 jugements:

$$\Gamma \vdash e \Rightarrow \tau \qquad \Gamma \vdash e \Leftarrow \tau$$

Règle habituelle: constructeurs vérifiés, éliminations inférées

Souvent suffisant pour éliminer annotations sauf sur `let`

Exprime un algorithme avec 2 fonctions mutuellements récursives:

$$\mathit{infer} : \mathit{Ctx} \to \mathit{Exp} \to \mathit{Typ}$$
$$\mathit{check} : \mathit{Ctx} \to \mathit{Exp} \to \mathit{Typ} \to \mathit{Unit}$$

Sémantique statique de Mini-ML pur

(*type ctx*) $\Gamma \quad ::= \quad \bullet \mid \Gamma, x : \tau$

$$\frac{\Gamma(x) = \tau}{\Gamma \vdash x \Rightarrow \tau} \qquad \frac{\Gamma \vdash e_1 \Rightarrow \tau_1 \to \tau_2 \quad \Gamma \vdash e_2 \Leftarrow \tau_1}{\Gamma \vdash e_1\ e_2 \Rightarrow \tau_2}$$

$$\frac{\Gamma, x : \tau_1 \vdash e \Leftarrow \tau_2}{\Gamma \vdash \lambda x \to e \Leftarrow \tau_1 \to \tau_2} \qquad \frac{\Gamma \vdash e \Leftarrow \tau}{\Gamma \vdash e : \tau \Rightarrow \tau} \qquad \frac{\Gamma \vdash e \Rightarrow \tau}{\Gamma \vdash e \Leftarrow \tau}$$

$$\frac{\Gamma \vdash e_1 \Leftarrow \texttt{Bool} \quad \Gamma \vdash e_2 \Leftarrow \tau \quad \Gamma \vdash e_3 \Leftarrow \tau}{\Gamma \vdash \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \Leftarrow \tau}$$

$$\frac{\Gamma \vdash e_1 \Rightarrow \tau_1 \quad \Gamma, x : \tau_1 \vdash e_2 \overset{a}{\Leftrightarrow} \tau_2}{\Gamma \vdash \texttt{let}\ x = e_1\ \texttt{in}\ e_2 \overset{a}{\Leftrightarrow} \tau_2}$$

Pseudocode de l'algorithme bidirectionnel

```
infer Γ e = case e
  | Evar x => lookup Γ x
  | Eapply e1 e2 => case infer Γ e1
    | Tarw t1 t2 => check Γ e2 t1; t2
  | Eannot e t => check Γ e t; t
  ...
check Γ e t = case e
  | Elam x e => case t
    | Tarw t1 t2 => check (extend Γ x t1) e t2
  | Eif e1 e2 e3 =>
    check Γ e1 Tbool; check Γ e2 t; check Γ e3 t
  ...
  | _ => assert (t = infer Γ e)
```

Sémantique dynamique de Mini-ML pur

(*values*) $v \quad ::= \quad \lambda x \to e \mid c\, v_1 \ldots v_n$

Réductions primitives:

$$(+)\, n_1\, n_2 \rightsquigarrow n_3 \qquad \text{where } n_3 = n_1 + n_2$$

$$(\lambda x{:}\tau \to e)\, v \rightsquigarrow e[v/x]$$

$$\texttt{let}\ x = v\ \texttt{in}\ e \rightsquigarrow e[v/x]$$

$$e : \tau \rightsquigarrow e$$

$$\texttt{if True then}\ e_2\ \texttt{else}\ e_3 \rightsquigarrow e_2$$

$$\texttt{if False then}\ e_2\ \texttt{else}\ e_3 \rightsquigarrow e_3$$

Sémantique dynamique de Mini-ML pur (suite)

Règles de congruence:

$$\frac{e_1 \rightsquigarrow e_1'}{e_1\ e_2 \rightsquigarrow e_1'\ e_2} \qquad \frac{e \rightsquigarrow e'}{v\ e \rightsquigarrow v\ e'}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\texttt{let}\ x = e_1\ \texttt{in}\ e_2 \rightsquigarrow \texttt{let}\ x = e_1'\ \texttt{in}\ e_2}$$

$$\frac{e_1 \rightsquigarrow e_1'}{\texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3 \rightsquigarrow \texttt{if}\ e_1'\ \texttt{then}\ e_2\ \texttt{else}\ e_3}$$

Ou aussi:

(*exps avec trou*) $E ::= \bullet \mid E\ e \mid v\ E \mid \texttt{let}\ x = E\ \texttt{in}\ e \mid \texttt{if}\ E\ \texttt{then}\ e_2\ \texttt{else}\ e_3$

Typage avec effets de bord

...

$$\frac{\Gamma \vdash e \Rightarrow \tau}{\Gamma \vdash \texttt{ref}\, e \Rightarrow \texttt{ref}\, \tau} \qquad \frac{\Gamma \vdash e \Rightarrow \texttt{ref}\, \tau}{\Gamma \vdash\, !e \Rightarrow \tau}$$

$$\frac{\Gamma \vdash e_1 \Leftarrow \texttt{Unit} \quad \Gamma \vdash e_2 \overset{a}{\Leftrightarrow} \tau_2}{\Gamma \vdash e_1; e_2 \overset{a}{\Leftrightarrow} \tau_2}$$

$$\frac{\Gamma \vdash e_1 \Rightarrow \texttt{ref}\, \tau \quad \Gamma \vdash e_2 \Leftarrow \tau}{\Gamma \vdash e_1 := e_2 \Rightarrow \texttt{Unit}}$$

Sémantique dynamique avec effets de bord?

$e \rightsquigarrow e'$ n'est plus suffisant, on a besoin de mémoire!

Nouvelles valeurs, renvoyées par $\texttt{ref}\ e$ et $e_1 := e_2$:

$$\text{(\textit{values})} \quad v ::= \ldots \mid () \mid \ell$$

$$\text{(\textit{heap})} \quad M ::= \bullet \mid M, \ell \mapsto v$$

Voyons d'abord une sémantique à grands pas:

$$(M; e) \downarrow (M'; v)$$

Évaluer e dans le tas M renvoie la valeur v avec un nouveau tas M'

Effets de bord à grands pas, partie pure

$$(M; v) \downarrow (M; v) \qquad \frac{(M; e) \downarrow (M'; v)}{(M; e : \tau) \downarrow (M'; v)}$$

$$\frac{(M; e_1) \downarrow (M_1; \lambda x \to e) \quad (M_1; e_2) \downarrow (M_2; v_2) \quad (M_2; e[v_2/x]) \downarrow (M'; v)}{(M; e_1\ e_2) \downarrow (M'; v)}$$

$$\frac{(M; e_1) \downarrow (M_1; v_1) \qquad (M_1; e_2[v_1/x]) \downarrow (M'; v)}{(M; \texttt{let}\ x = e_1\ \texttt{in}\ e_2) \downarrow (M'; v)}$$

$$\frac{(M; e_1) \downarrow (M_1; v_1) \qquad (M_1; e) \downarrow (M'; v) \qquad e = \begin{cases} e_2 & \text{if } v_1 = \texttt{True} \\ e_3 & \text{if } v_1 = \texttt{False} \end{cases}}{(M; \texttt{if}\ e_1\ \texttt{then}\ e_2\ \texttt{else}\ e_3) \downarrow (M'; v)}$$

¡L'existence de $:=$ oblige à changer les règles pures!

Effets de bord à grands pas, partie impure

$$\frac{(M; e) \downarrow (M'; v) \qquad \ell \text{ fresh}}{(M; \texttt{ref}\ e) \downarrow (M', \ell \mapsto v; \ell)}$$

$$\frac{(M; e) \downarrow (M'; \ell) \qquad M'(\ell) = v}{(M; !e) \downarrow (M'; v)}$$

$$\frac{(M; e_1) \downarrow (M_1; \ell) \qquad (M_1; e_2) \downarrow (M'; v)}{(M; e_1 := e_2) \downarrow (M', \ell \mapsto v; ())}$$

$$\frac{(M; e_1) \downarrow (M_1; v_1) \qquad (M_1; e_2) \downarrow (M'; v)}{(M; e_1; e_2) \downarrow (M'; v))}$$

Des expressions et des valeurs

Le règle d'évaluation des valeurs est formellement incorrecte

$$(M; v) \downarrow (M; v)$$

Parce qu'à gauche on doit avoir une *expression*:

- Les valeurs v ne sont plus un sous-ensemble des expressions e!

Il faut $(M; c) \downarrow (M; c)$, $(M; \lambda x \to e) \downarrow (M; \lambda x \to e)$, ...

Pour les petits pas, ça ne suffit pas; il faut étendre e:

ou même

$$\text{(exps)} \quad e \quad ::= \quad ... \mid \ell$$

$$\text{(exps)} \quad e \quad ::= \quad ... \mid v$$

Effets de bord à petits pas, partie pure

Réductions primitives:

$$(M; (+)\ n_1\ n_2) \rightsquigarrow (M; n_3) \quad \text{where } n_3 = n_1 + n_2$$

$$(M; (\lambda x : \tau \to e)\ v) \rightsquigarrow (M; e[v/x])$$

$$(M; \texttt{let}\ x = v\ \texttt{in}\ e) \rightsquigarrow (M; e[v/x])$$

$$(M; e : \tau) \rightsquigarrow (M; e)$$

$$(M; \texttt{if True then}\ e_2\ \texttt{else}\ e_3) \rightsquigarrow (M; e_2)$$

$$(M; \texttt{if False then}\ e_2\ \texttt{else}\ e_3) \rightsquigarrow (M; e_3)$$

Rien de changé, mis à part l'ajout du M partout

Effets de bord à petits pas, partie impure

Réductions primitives:

$$(M; \texttt{ref}\ v) \rightsquigarrow (M, \ell \mapsto v; \ell) \quad \text{where } \ell\ \texttt{fresh}$$

$$(M; !\ell) \rightsquigarrow (M; v) \quad \text{where } M(\ell) = v$$

$$(M; \ell := v) \rightsquigarrow (M, \ell \mapsto v; ())$$

$$(M; (); e) \rightsquigarrow (M; e)$$

(*exps avec trou*) $E ::= \bullet \mid E\ e \mid v\ E \mid \texttt{let}\ x = E\ \texttt{in}\ e$

$\mid \texttt{if}\ E\ \texttt{then}\ e_2\ \texttt{else}\ e_3$

$\mid \texttt{ref}\ E \mid !E \mid E := e \mid v := E \mid E; e$

Règles de typage valides?

Comment s'assurer de la validité des règles de typage?

Si $e : \tau$, la valeur v retournée par e devrait avoir type τ

La *cohérence* (*soundness*) des règles peut se décomposer en:

- *préservation des types*: Si $e : \tau$ et $e \rightsquigarrow e'$ alors $e' : \tau$
- *progrès*: Si $e : \tau$, ou $e \rightsquigarrow e'$ ou e est une valeur

Ensemble, cela donne:

Si $e : \tau$ et $e \rightsquigarrow^* e'$, alors e' n'est pas *stuck*:

ou $e' \rightsquigarrow e''$ ou e' est une valeur

"well-typed programs do not go wrong"

Cohérence du typage de Mini-ML?

En général, *progrès* ne s'applique que dans un environnement vide:

Si $\bullet \vdash e : \tau$, alors ou $(M; e) \rightsquigarrow (M'; e')$ ou e est une valeur

Préservation des types implique, par exemple:

Vu que $\bullet \vdash \texttt{ref}\ 4 : \texttt{ref Int}$ et $(M; \texttt{ref}\ 4) \rightsquigarrow (M, \ell \mapsto 4; \ell)$

alors $\bullet \vdash \ell : \texttt{ref Int}$

Il nous faut donc une règle de typage pour ℓ: $\dfrac{?}{\Gamma \vdash \ell : \tau}$

Ψ donnera un type à chaque adresse:

(*heap type*) $\Psi \quad ::= \quad \bullet \mid \Psi, \ell : \tau$

Nouveaux jugements de typage

Il faut vérifier que Ψ est bien le type de M, avec $\vdash M : \Psi$:

$$\frac{\Psi \vdash M : \Psi}{\vdash M : \Psi} \qquad \frac{}{\Psi \vdash \bullet : \bullet} \qquad \frac{\Psi \vdash M : \Psi' \qquad \Psi; \bullet \vdash v \Leftarrow \tau}{\Psi \vdash M, \ell \mapsto v : \Psi', \ell : \tau}$$

Et les jugements prennent maintenant la forme: $\Psi; \Gamma \vdash e \overset{a}{\Leftrightarrow} \tau$

$$\frac{\Gamma(x) = \tau}{\Psi; \Gamma \vdash x \Rightarrow \tau} \qquad \ldots \qquad \frac{\Psi(\ell) = \tau}{\Psi; \Gamma \vdash \ell \Rightarrow \texttt{ref}\ \tau}$$

Aucun jugement ne modifie Ψ, e.g.:

$$\frac{\Psi; \Gamma \vdash e \Rightarrow \tau}{\Psi; \Gamma \vdash \texttt{ref}\ e \Rightarrow \texttt{ref}\ \tau} \qquad \frac{\Psi; \Gamma \vdash e_1 \Rightarrow \texttt{ref}\ \tau \qquad \Psi; \Gamma \vdash e_2 \Leftarrow \tau}{\Psi; \Gamma \vdash e_1 := e_2 \Rightarrow \texttt{Unit}}$$

Cohérence du typage de Mini-ML

Progrès:

$$\text{Si} \quad \vdash M : \Psi \quad \wedge \quad \Psi; \bullet \vdash e \Leftarrow \tau$$

$$\text{alors} \quad (M; e) \rightsquigarrow (M'; e') \quad \vee \quad e \text{ est une valeur}$$

Préservation des types:

$$\text{Si} \quad \vdash M : \Psi \quad \wedge \quad \Psi; \Gamma \vdash e \Leftarrow \tau \quad \wedge \quad (M; e) \rightsquigarrow (M'; e')$$

$$\text{alors} \quad \exists \Psi'. \quad \vdash M' : \Psi' \quad \wedge \quad \Psi'; \Gamma \vdash e' \Leftarrow \tau$$

Preuve de préservation des types

Dans le cas où $e = (\lambda x : \tau \to e_1)\ v$ et $e' = e_1[v/x]$,

on doit prouver $\Psi; \Gamma \vdash e_1[v/x] \Leftarrow \tau_2$,

sachant que $\Psi; \Gamma, x : \tau_1 \vdash e_1 \Leftarrow \tau_2$ et $\Psi; \Gamma \vdash v \Leftarrow \tau_1$

Lemme de substitution ressemble habituellement à:

Si $\Psi; \Gamma, x : \tau_1, \Gamma' \vdash e : \tau_2 \quad \wedge \quad \Psi; \Gamma \vdash v : \tau_1$

alors $\Psi; \Gamma, \Gamma' \vdash e[v/x] : \tau_2$

Ce genre de lemme est récurrent dans les preuves de préservation

Preuve de progrès

Dans le cas $e = e_1\ e_2$, si e_1 ou e_2 n'est pas une valeur,

alors on trouve immédiatement un pas $e \rightsquigarrow e'$ par induction,

sinon il nous faut prouver que $e = (\lambda x \to e_3)\ e_2$ pour appliquer β

Lemme de formes canoniques ressemble habituellement à:

$$\text{Si} \quad \Psi; \bullet \vdash v : \tau_1 \to \tau_2$$

$$\text{alors} \quad v = \lambda x \to e$$

Ce genre de lemme est récurrent dans les preuves de progrès