Autres langages

- Langages fonctionnels — Lisp, ML, Haskell

  C'est ce qu'on va étudier dans la suite,

  notamment pour des raisons de simplicité.

- Langages logiques — Prolog
- Langages à objets — Java, C++
- Langages pour le calcul numérique — Fortran90

  (parallélisme, calcul vectoriel)

- Langages de bases de données — SQL, O2
- etc.

Langages fonctionnels Calcul par appels de fonctions

			Termes (fonctions, données)
$M, N, P, \ldots$	$::=$	x	variables (non modifiables)
	$\mid$	$M\,N$	application de M à N
	$\mid$	$\texttt{fun}\ x \to M$	fonction qui à x associe M
	$\mid$	$\texttt{let}\ x = M\ \texttt{in}\ N$	définition
	$\mid$	$\texttt{ref}\ M \mid !M \mid M{:=}N$	références (modifiables)
	$\mid$	$\texttt{0} \mid \texttt{1} \mid \ldots \mid M{+}N \mid \ldots$	arithmétique
	$\mid$	$(M_1, \ldots, M_n) \mid \pi_i M$	n-uplets
	$\mid$	$M; N$	séquence
	$\mid$	$\texttt{if}\ M\ \texttt{then}\ N\ \texttt{else}\ P$	tests
$Prog$	$::=$	$(\texttt{letrec}\ x = M;;)^*$	Programmes

Comment compiler ça? Il faut en connaître la sémantique...

Exemple: `cat` en fonctionnel

```
letrec boucle_interne = fun f ->
       let c = fgetc f
       in if c=EOF
             then 0
          else (putchar c; boucle_interne f);;
letrec boucle_externe = fun i -> fun argc -> fun argv ->
       if i<argc
          then let nom = sub argv i
            in let f = fopen nom "r"
            in (boucle_interne f; fclose f;
                boucle_externe (i+1) argc argv)
       else 0;;
letrec main = fun argc -> fun argv ->
       boucle_externe 1 argc argv;;
```

Sémantiques pour un langage fonctionnel jouet

- Un environnement $\rho \in Env =_{def} Var \to Val$;
- Une mémoire $\mu \in Mem =_{def} Addr \to Val$;

- Valeurs $V \in Val$:
  $Val ::= Int_{32} | Addr | Val^* | Val \to Val$

$$Int_{32} ::= [-\mathtt{0x80000000}..\mathtt{0x7fffffff}]$$

(valeurs=entiers, adresses, n-uplets, ou fonctions)

Exercice: il y a plusieurs problèmes avec cette définition: lesquels?

- Des jugements:

$$\rho, \mu \vdash M \Rightarrow \mu', V$$

(c'est une sémantique opérationnelle à grands pas.)

Sémantique d'appel par valeur gauche-droite: 1er essai

$$\frac{}{\rho, \mu \vdash x \Rightarrow \mu, \rho(x)}\ (Var)$$

$$\frac{\rho, \mu \vdash M \Rightarrow \mu', f \in (Val \to Val) \quad \rho, \mu' \vdash N \Rightarrow \mu'', V}{\rho, \mu \vdash MN \Rightarrow \mu'', f(V)}\ (App)$$

Exercice: quels effets de bord f est-elle autorisée à effectuer? Proposer un remède.

Sémantique ...: 2ème essai

On change: $Val ::= Int_{32} | Addr | Val^* | Mem \times Val \to Mem \times Val$

$$\frac{}{\rho, \mu \vdash x \Rightarrow \mu, \rho(x)} \; (Var)$$

$$\frac{\rho, \mu \vdash M \Rightarrow \mu', f \in (Val \to Val) \quad \rho, \mu' \vdash N \Rightarrow \mu'', V \qquad f(\mu'', V) = (\mu''', V')}{\rho, \mu \vdash MN \Rightarrow \mu''', V'} \; (App)$$

$$\frac{\forall V \in Val, \mu' \in Mem \cdot \rho[x := V], \mu' \vdash M \Rightarrow f(\mu', V)}{\rho, \mu \vdash \texttt{fun}\ x \to M \Rightarrow \mu, f} \; (Fun)$$

Exercice: et si M ne termine pas sur V, μ'?

Sémantique ...: comment faire maintenant?

Deux solutions standard aux problèmes posés par les fonctions:

- Trouver une interprétation non-standard de la notion de fonction, dans laquelle

$$Val ::= Int_{32} | Addr | Val^* | Mem \times Val \to Mem \times Val$$

  a une solution (avec $Mem = Addr \to Val$).

  théorie des domaines (Dana Scott)

  gère aussi la non-terminaison!

  a été très fertile dans le domaine des langages de programmation.

- Éluder le problème, et représenter les fonctions comme elles seront compilées (par une adresse dans le code, voir transparent suivant).

- Bien sûr, la deuxième solution est correcte par rapport à la première (théorème de raffinement)... mais ça n'est pas si évident.

Comment représenter les fonctions?

- Première approche: comme en C, une fonction=une adresse dans le code.
- Défaut:

```
letrec f = fun x ->
               let g = fun y -> x+y
               in g;;
letrec a = f 3;;
```

  Que retourne `a`? L'adresse du code pour `g`? Mais alors que vaut `x` dans ce bout de code?

  Exercice: comment peut-on réparer ce problème?

Les clôtures

Slogan: une fonction = une adresse dans le code + un environnement.

Sémantiquement: $$Val ::= Int_{32} | Addr | Val^* | \underbrace{Var \times Terme \times Env}_{\text{clôture}}$$

$$\frac{}{\rho, \mu \vdash x \Rightarrow \mu, \rho(x)}\,(Var)$$

$$\frac{\rho, \mu \vdash M \Rightarrow \mu', \overbrace{(x, M', \rho')}^{\text{clôture}} \quad \rho, \mu' \vdash N \Rightarrow \mu'', V \quad \rho'[x := V], \mu'' \vdash M' \Rightarrow \mu''', V'}{\rho, \mu \vdash M N \Rightarrow \mu''', V'}\,(App)$$

$$\frac{}{\rho, \mu \vdash \texttt{fun}\ x \to M \Rightarrow \mu, \underbrace{(x, M, \rho)}_{\text{clôture}}}\,(Fun)$$

Sémantique: le reste

$$\frac{\rho, \mu \vdash M \Rightarrow \mu', V \quad a \in Addr \setminus \mathrm{dom}\mu'}{\rho, \mu \vdash \texttt{ref}\ M \Rightarrow \mu' \oplus \{a \mapsto V\}, V} (Ref)$$

$$\frac{\rho, \mu \vdash M \Rightarrow \mu', a \in \mathrm{dom}\mu'}{\rho, \mu \vdash !M \Rightarrow \mu', \mu'(a)} (Bang)$$

$$\frac{\rho, \mu \vdash M \Rightarrow \mu', a \in \mathrm{dom}\mu' \quad \rho, \mu' \vdash N \Rightarrow \mu'', V}{\rho, \mu \vdash M{:=}N \Rightarrow \mu''[a \mapsto V], V} (Assign)$$

Exercice: décrire la sémantique de `let`, 0, $+$, `if`, etc.