Dépendances et linéarité

Effets de bord $\Rightarrow$ casse tout:

$$\mathit{refl}\,(\mathit{random}\ 10) : \mathit{random}\ 10 = \mathit{random}\ 10$$

Effets de bord = dépendances implicites sur “world”:

$$\mathit{random} : \mathit{Nat} \to \mathit{World} \to (\mathit{Nat} \times \mathit{World})$$

Mais on ne peut pas manipuler le monde à sa guise:

$$(n, \mathit{newWorld1}) = \mathit{launchMissiles}\ \mathit{oldWorld}$$

$$(n, \mathit{newWorld2}) = \mathit{read}\ \texttt{"Guardian"}\ \mathit{oldWorld}$$

World est le type d’une *ressource*, i.e. non-duplicable

Taming effects

Logique linéaire

- Contrôler l'usage des variables contenant des ressources

CPS (Continuation Passing Style)

- Cacher les effets dans les continuations

Monads

- Choisir avec soin des primitives et des combinateurs

Effets algébriques

- Sorte de généralisation des exceptions

Logique linéaire

Inventée par J-Y Girard (comme System F)

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

$$\frac{\Gamma_1 \vdash e_1 : \tau_1 \multimap \tau_2 \qquad \Gamma_2 \vdash e_2 : \tau_1}{\Gamma_1, \Gamma_2 \vdash e_1\ e_2 : \tau_2}$$

Pas de *weakening*

Le contexte décrit la consommation de ressources

Conjonction additive: $\tau_1 \& \tau_2$

CPS (Continuation Passing Style)

Au lieu du simple type α, une expression avec effet reçoit type:

$$\neg\neg\alpha \quad \simeq \quad (\alpha \to \bot) \to \bot$$

On a donc plus accès direct à la valeur de type α

Les appels par continuations imposent un ordonnancement

La *conversion CPS* transforme $f\ (e_1,\ e_2)$ en:

$$f : (\tau_1 \times \tau_2) \to \tau_3$$
$$[\![f]\!] : (\tau_1 \times \tau_2 \times (\tau_3 \to \bot)) \to \bot$$
$$[\![f\ (e_1,\ e_2)]\!] = \lambda k.[\![e_1]\!]\ (\lambda x_1.[\![e_2]\!]\ (\lambda x_2.[\![f]\!]\ (x_1, x_2, k)))$$

Un langage en style CPS

$$v ::= c \mid x \mid \lambda \vec{x}.e$$

$$p ::= + \mid - \mid \texttt{cons} \mid \ldots$$

$$e ::= \texttt{let}\, x = p(\vec{v})\, \texttt{in}\, e \mid v\,(\vec{v})$$

Distinction entre les *valeurs* et les *expressions*

Appels de fonctions toujours terminaux

Pile? Quelle pile?

Monads

Cacher le passage de ressources derrière un type *opaque*:

$$\texttt{type}\ \mathit{IO}\,\alpha = \mathit{World} \to (\alpha \times \mathit{World})$$

Exporter des primitives:

$$\mathit{random} : \mathit{Int} \to \mathit{IO}\ \mathit{Int}$$

$$\ldots$$

Et de quoi les combiner sans casser la linéarité:

$$\mathit{return} : \alpha \to \mathit{IO}\,\alpha$$

$$\mathit{bind} : \mathit{IO}\,\alpha \to (\alpha \to \mathit{IO}\,\beta) \to \mathit{IO}\,\beta$$

Histoire des monads

Introduits dans Haskell pour gérer les effets de bord

Wildly popular pour structurer le code

Développement d'autres monads plus "précis"

- Limités à une catégorie d'effets

Besoin de combiner/composer divers monads

- Pas possible en général!