Projet de recherche

Résumé de l'article **Handling Algebraic effects**

Jordan Ischard

Université d'Orléans

2020-2021

Sommaire

Sommaire

Ajout de principes impératifs dans un langage fonctionnel

But

Ajouter des fonctionnalités en plus dans les langages fonctionnels

Ajout de principes impératifs dans un langage fonctionnel

But

Ajouter des fonctionnalités en plus dans les langages fonctionnels

Problème

Ajoute des effets de bords que les langages fonctionnels purs non pas.

Ajout de principes impératifs dans un langage fonctionnel

But

Ajouter des fonctionnalités en plus dans les langages fonctionnels

Problème

Ajoute des effets de bords que les langages fonctionnels purs non pas.

Exemple

On peut avoir des effets de bords pour les appels mémoires ou encore les entrées/sorties.

Ajout de principes impératifs dans un langage fonctionnel

But

Ajouter des fonctionnalités en plus dans les langages fonctionnels

Problème

Ajoute des effets de bords que les langages fonctionnels purs non pas.

Exemple

On peut avoir des effets de bords pour les appels mémoires ou encore les entrées/sorties.

Idée

Créer une structure qui va gérer ces effets.

Sommaire

Réponses déjà proposées

Les Monades

Eugenio Moggi a proposé le principe de Monade pour gérer les effets.

Réponses déjà proposées

Les Monades

Eugenio Moggi a proposé le principe de Monade pour gérer les effets.

Les Monades en Haskell

```
1 instance Monad Maybe where
2   return x = Just x
3
4   Nothing >>= f = Nothing
5   Just x >>= f  = f x
6
7   fail _ = Nothing
8
```

Réponses déjà proposées

Les Monades

Eugenio Moggi a proposé le principe de Monade pour gérer les effets.

Les Monades en Haskell

```haskell
instance Monad Maybe where
  return x = Just x

  Nothing >>= f = Nothing
  Just x >>= f  = f x

  fail _ = Nothing

```

Les théories d'équations

Plotkin et Power ont proposé des opérations comme source des effets et une théorie d'équation pour décrire leurs propriétés.

Sommaire

Sommaire

Comment gérer les effets ?

Intuition

Reprendre le principe de Plotkin et Power et l'associer aux travaux de Benton et Kennedy sur les gestionnaires.

Comment gérer les effets ?

Intuition

Reprendre le principe de Plotkin et Power et l'associer aux travaux de Benton et Kennedy sur les gestionnaires.

Principe

Les effets vont avoir pour source des opérations et vont être géré par un tableaux associatifs (map) contenu dans un gestionnaire.

Sommaire

Qu'ajoute-on pour mettre en place la gestion des effets ?

Syntaxe ajouté

On part de la syntaxe proposé par Levy dans son article sur *call-by-push-value*.

$$\begin{aligned} \textit{source de l'effet} &: \mathbf{op}_V(x : \beta.M') \\ \textit{structure de gestion} &: M \ \mathbf{handled\ with}\ H\ \textit{to}\ x : A.N \\ \textit{gestionnaire} &: \{\mathbf{op}_{z:\alpha}(k : \beta \to \underline{C}) \mapsto M_{\mathbf{op}}\}_{\mathbf{op}:\alpha\to\beta} \end{aligned}$$

Qu'ajoute-on pour mettre en place la gestion des effets ?

Syntaxe ajouté

On part de la syntaxe proposé par Levy dans son article sur *call-by-push-value*.

$$\begin{aligned} \textit{source de l'effet} &: \mathbf{op}_V(x : \beta.M') \\ \textit{structure de gestion} &: M \textbf{ handled with } H \textit{ to } x : A.N \\ \textit{gestionnaire} &: \{\mathbf{op}_{z:\alpha}(k : \beta \to \underline{C}) \mapsto M_{\mathbf{op}}\}_{\mathbf{op}:\alpha\to\beta} \end{aligned}$$

Fonctionnement

On reprend la strucutre de gestion ci-dessus avec $\mathbf{op}_V(y.M') \in M$ et $\{\mathbf{op}_z(k) \mapsto M_{\mathbf{op}}\} \in H$.

$$M_{\mathbf{op}}[V/z, M'[W/y] \textbf{ handled with } H \textbf{ to } x : A.N/k(W)]$$

Sommaire

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{l:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(n)\}$$

Prenons l'expression suivante :

$$\begin{aligned}
&\mathbf{let}\ n : \mathbf{nat}\ \mathbf{be}\ 20\ \mathbf{in}\\
&\mathbf{get}_l(x : \mathbf{nat}.get_l(y : \mathbf{nat}.\mathbf{return}\ x + y))\\
&\mathbf{handled\ with}\ H_{temporary}\ \mathbf{to}\ z : A.\mathbf{return}\ z + 2
\end{aligned}$$

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{l:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(n)\}$$

Prenons l'expression suivante :

$$\begin{aligned}&\mathbf{let}\ n : \mathbf{nat}\ \mathbf{be}\ 20\ \mathbf{in}\\ &\mathbf{get}_l(x : \mathbf{nat}.get_l(y : \mathbf{nat}.\mathbf{return}\ x + y))\\ &\mathbf{handled\ with}\ H_{temporary}\ \mathbf{to}\ z : A.\mathbf{return}\ z + 2\end{aligned}$$

La réponse

C'est 42 !

Sommaire

Sommaire

Les gestionnaires en général sont indécidable

TODO : Trop complexe, obliger de décider entre laisser l'utilisateur faire tout en partant du principe qu'il peut se louper ou le contraindre

Sommaire

Gestionnaire décidable sous certaines conditions

TODO : principe de gestionnaire simple, de famille uniformément simple etc.

Sommaire

Sommaire

Motivation et fonctionnement divergent

TODO : motivation, source des effets etc.

Sommaire

Passage du langage de l'article vers **erpl**

TODO : l'exemple en gros

Sommaire

Une histoire d'appels systèmes

TODO : expliquer le problème de l'appel système et du gestionnaire global implicite

Sommaire

Sample frame title

In this slide, some important text will be highlighted because it's important. Please, don't abuse it.

Remark

Sample text

Important theorem

Sample text in red box

Examples

Sample text in green box. The title of the block is "Examples".