Projet de recherche

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

Jordan Ischard

Université d'Orléans

2020-2021

Sommaire

Introduction

- Problème de l'impératif dans un langage fonctionnel
- Proposition pour gérer les effets

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

Conclusion

Sommaire

Introduction

- Problème de l'impératif dans un langage fonctionnel
- Proposition pour gérer les effets

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

Conclusion

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

Introduction

- Problème de l'impératif dans un langage fonctionnel
- Proposition pour gérer les effets

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

Conclusion

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

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

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

  fail _ = Nothing

```

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

Introduction

Gestion des effets dans l'article
- Intuition et principe
- Mise en place dans une syntaxe
- Exemple

Validé des gestionnaires

Confrontation

Conclusion

Sommaire

Introduction

Gestion des effets dans l'article

- Intuition et principe
- Mise en place dans une syntaxe
- Exemple

Validé des gestionnaires

Confrontation

Conclusion

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

Introduction

Gestion des effets dans l'article

- Intuition et principe
- Mise en place dans une syntaxe
- Exemple

Validé des gestionnaires

Confrontation

Conclusion

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

Introduction

Gestion des effets dans l'article

- Intuition et principe
- Mise en place dans une syntaxe
- Exemple

Validé des gestionnaires

Confrontation

Conclusion

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 :

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

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 :

let n : **nat be** 20 **in**

$\mathbf{get}_l(x : \mathbf{nat}.\mathbf{get}_l(y : \mathbf{nat}.\mathbf{return}\ x + y))$

handled with $H_{temporary}$ **to** $z : A.\mathbf{return}\ z + 2$

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{y:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(20)\}$$

Prenons l'expression suivante :

$$\mathbf{get}_l(x : \mathbf{nat}.\mathbf{get}_l(y : \mathbf{nat}.\mathbf{return}\ x + y))$$
$$\mathbf{handled\ with}\ H_{temporary}\ \mathbf{to}\ z : A.\mathbf{return}\ z + 2$$

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{y:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(20)\}$$

Prenons l'expression suivante :

$$\mathbf{get}_{l}(y : \mathbf{nat}.\mathbf{return}\ 20 + y)$$
$$\mathbf{handled\ with}\ H_{temporary}\ \mathbf{to}\ z : A.\mathbf{return}\ z + 2$$

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{y:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(20)\}$$

Prenons l'expression suivante :

return $20 + 20$

handled with $H_{temporary}$ **to** $z : A.$**return** $z + 2$

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{y:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(20)\}$$

Prenons l'expression suivante :

$$\mathbf{return}\ 40 + 2$$

dérogation à la lecture seule

Valeur Temporaire sans modification dans la mémoire

Le gestionnaire définit pour l'effet est le suivant :

$$H_{temporary} = \{\mathbf{get}_{y:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(20)\}$$

Prenons l'expression suivante :

$$\mathbf{return}\ 40 + 2$$

La réponse

C'est 42 !

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires
- Gestionnaire générique indécidable
- Sous-ensemble décidable

Confrontation

Conclusion

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

- Gestionnaire générique indécidable
- Sous-ensemble décidable

Confrontation

Conclusion

Les gestionnaires en général sont indécidable

Limite des gestionnaires

En général la validité des gestionnaires est indécidable.

Les gestionnaires en général sont indécidable

Limite des gestionnaires

En général la validité des gestionnaires est indécidable.

Comment palier à se problème

Deux approches sont possibles :

1. Obliger l'utilisateur a utiliser des gestionnaires que l'on sait valide (donc prédéfinis dans le langage) : la responsabilité est au créateur du langage
2. Laisser l'utilisateur faire comme il le souhaite tout en lui laissant la possibiliter de faire des gestionnaires invalide : la responsabilité est à l'utilisateur.

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

- Gestionnaire générique indécidable
- Sous-ensemble décidable

Confrontation

Conclusion

Gestionnaire décidable sous certaines conditions

Vers des gestionnaires plus simple

Il est possible de trouver un sous-ensemble de gestionnaire qui est décidable.

Gestionnaire décidable sous certaines conditions

Vers des gestionnaires plus simple

Il est possible de trouver un sous-ensemble de gestionnaire qui est décidable.

Il faut les conditions suivantes :

1. Une signature d'opérations simple

Gestionnaire décidable sous certaines conditions

Vers des gestionnaires plus simple

Il est possible de trouver un sous-ensemble de gestionnaire qui est décidable.

Il faut les conditions suivantes :

1. Une signature d'opérations simple
2. TODO

Gestionnaire décidable sous certaines conditions

Vers des gestionnaires plus simple

Il est possible de trouver un sous-ensemble de gestionnaire qui est décidable.

Il faut les conditions suivantes :

1. Une signature d'opérations simple
2. TODO
3. TODO

Gestionnaire décidable sous certaines conditions

Vers des gestionnaires plus simple

Il est possible de trouver un sous-ensemble de gestionnaire qui est décidable.

Il faut les conditions suivantes :

1. Une signature d'opérations simple
2. TODO
3. TODO

Gestionnaire simple

On parle de gestionnaire simple et grâce à une réécriture des termes avec des patrons on peut même parler de famille de gestionnaire uniformément simple.

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation
- Les divergences
- Exemple de conversion entre langages
- Difficulté d'implémentation

Conclusion

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

- Les divergences
- Exemple de conversion entre langages
- Difficulté d'implémentation

Conclusion

Motivation et fonctionnement divergent

Langage de l'article	Langage **erpl**
◦ Créer pour voir l'aspect mathématique de la gestion d'effets algébriques	◦ Créer pour avoir une version allégé du langage Ocaml

Motivation et fonctionnement divergent

Langage de l'article	Langage **erpl**
◦ Créer pour voir l'aspect mathématique de la gestion d'effets algébriques	◦ Créer pour avoir une version allégé du langage Ocaml
◦ La source d'un effet est une opération	◦ La source d'un effet est un type

Motivation et fonctionnement divergent

Langage de l'article	Langage **erpl**
◦ Créer pour voir l'aspect mathématique de la gestion d'effets algébriques	◦ Créer pour avoir une version allégé du langage Ocaml
◦ La source d'un effet est une opération	◦ La source d'un effet est un type
◦ Activation implicite de l'effet	◦ Activation explicite de l'effet

Motivation et fonctionnement divergent

Langage de l'article	Langage **erpl**
◦ Créer pour voir l'aspect mathématique de la gestion d'effets algébriques	◦ Créer pour avoir une version allégé du langage Ocaml
◦ La source d'un effet est une opération	◦ La source d'un effet est un type
◦ Activation implicite de l'effet	◦ Activation explicite de l'effet
◦ Gestionnaire global implicite car l'activation est implicite	◦ Pas de gestionnaire global, tout activation sans gestionnaire provoque une erreur

Table: Comparaison entre les deux langages

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

- Les divergences
- Exemple de conversion entre langages
- Difficulté d'implémentation

Conclusion

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

Reprise de l'exemple précédent

let n : **nat be** 20 **in**
$\mathbf{get}_l(x : \mathbf{nat}.\mathbf{get}_l(y : \mathbf{nat}.\mathbf{return}\ x + y))$
handled with $\{\mathbf{get}_{l:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(n)\}$ **to** $z : A.\mathbf{return}\ z{+}2$

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

Reprise de l'exemple précédent

let n : **nat be** 20 **in**
$\mathbf{get}_l(x : \mathbf{nat}.\mathbf{get}_l(y : \mathbf{nat}.\mathbf{return}\ x + y))$
handled with $\{\mathbf{get}_{l:\mathbf{loc}}(k : \mathbf{nat} \to \underline{C}) \mapsto k(n)\}$ **to** $z : A.\mathbf{return}\ z{+}2$

Reprise de l'exemple converti en **erpl**

```
1  type memory = Get of 'a ref -> 'a
2               | Set of 'a ref * 'a -> unit;;
3
4  let get = Get(fun x -> !x) and n = 20 in
5            handle ((fun x -> fun y -> x + y + 2)
6                    (perform (get,l))) (perform (get,l))
7            with (Get f,x),k -> k n ;;
8
```

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

- Les divergences
- Exemple de conversion entre langages
- Difficulté d'implémentation

Conclusion

Une histoire d'appels systèmes

Comment ça fonctionne ?

En prenant, le principe d'opération comme source d'effet on implique une gestion des opérations particuliers.
Quand on a pas de gestionnaire, les opérations se *gèrent elles-même*.

Une histoire d'appels systèmes

Comment ça fonctionne ?

En prenant, le principe d'opération comme source d'effet on implique une gestion des opérations particuliers.
Quand on a pas de gestionnaire, les opérations se *gèrent elles-même*.

Idée

Créer un gestionnaire globale implicite ajouté à la compilation.

Une histoire d'appels systèmes

Comment ça fonctionne ?

En prenant, le principe d'opération comme source d'effet on implique une gestion des opérations particuliers.
Quand on a pas de gestionnaire, les opérations se *gèrent elles-même*.

Idée

Créer un gestionnaire globale implicite ajouté à la compilation.

Problèmes d'appels systèmes

Cela veut dire que notre gestionnaire globale doit être capable de faire des appels systèmes et donc que le langage en soit capable.
Pas facilement intégrable !

Sommaire

Introduction

Gestion des effets dans l'article

Validé des gestionnaires

Confrontation

Conclusion

Que peut-on conclure ?

TODO : No se