EFEKTY ALGEBRAICZNE I ICH HANDLERY

Maciej Piróg

@fp-wroc, 3.10.2018, Wrocław

Projekt finansowany ze środków przeznaczonych na program finansowania badań naukowych i innowacji UE "Horyzont 2020" na podstawie umowy Nr 665778 o dofinansowanie działań "Marie Skłodowska-Curie"

KIM JESTEM?

Maciej Piróg
http://www.ii.uni.wroc.pl/~mpirog

obecnie:
- Instytut Informatyki, Uniwersytet Wrocławski

wcześniej:
- Uniwersytet w Leuven, Belgia
- Uniwerystet Oksfordzki, UK

ostatnio programuję w:
- Helium, Haskell, Javascript, Racket, TEX

CO TU ROBIĘ?

Projekt Efekty Algebraiczne i Kontynuacje realizowany na UWr

finansowany ze środków przeznaczonych na program finansowania badań naukowych i innowacji UE "Horyzont 2020"

ZA CO KOCHAMY FP?

Bo możemy programować tak:

Wejście --[FUNKCJA]--> Wyjście

Ale czy na pewno zawsze tak możemy/chcemy?

EFEKTY

Czasem musiamy/chcemy mieć dostęp do:

* Obsługa błędów (np. wyjątki)
* Mutowalny stan
* Wejście-wyjście (wyjątkowo przydatne!)
* Generator liczb losowych
* Niedeterminizm (np. backtracking)
* Wielowątkowość
* Stałe globalne (np.parametry wejściowe)

Checemy robić efekty, ale chcemy robić je z głową!!

WSPARCIE OD JĘZYKA / KONWENCJONALNYCH ABSTRAKCJI

* Na pałę (LISP, OCaml, SML)
* Monady (Haskell)
* ...??

WSPARCIE OD JĘZYKA / KONWENCJONALNYCH ABSTRAKCJI

* Na pałę (LISP, OCaml, SML)
  - Brak kontroli nad semantyką (np. przy używaniu wielu efektów jednocześnie)
  - Brak sensownego wsparcia ze strony systemu typów
  - Słabo z tworzeniem własnych efektów

* Monady (Haskell)

* ...??

WSPARCIE OD JĘZYKA / KONWENCJONALNYCH ABSTRAKCJI

* Na pałę (LISP, OCaml, SML)

* Monady (Haskell)
  - Blady strach
  - Niefunkcyjny styl programowania (“Haskell to najlepszy imperatywny język programowania na świecie” - Bob Harper)
  - Trudna modularność w efektach

* ...??

WSPARCIE OD JĘZYKA / KONWENCJONALNYCH ABSTRAKCJI

* Na pałę (LISP, OCaml, SML)

* Monady (Haskell)

* ...??
  - Funkcyjny styl programowania?
  - Rozszerzalna semantyka?
  - Wsparcie od strony systemu typów?
  - Modularność w efektach?

WSPARCIE OD JĘZYKA / KONWENCJONALNYCH ABSTRAKCJI

* Na pałę (LISP, OCaml, SML)

* Monady (Haskell)

* ...??
  - Funkcyjny styl programowania?
  - Rozszerzalna semantyka?
  - Wsparcie od strony systemu typów?
  - Modularność w efektach?

WYDAJE SIĘ, ŻE TAK!!!

EFEKTY ALGEBRAICZNE

* Pochodzą z głębin teoretycznych, ale nie trzeba zanurzyć nawet małego paluszka

* Kilka eksperymentalnych języków: Eff, Frank, Koka, Links, Helium

PRZYKŁAD: WYJĄTKI

```
try
   ##############################
   ##############################
   ######### throw exc ##########
   ##############################
catch
   exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try
                    ##################
   ############################
   ######### throw exc ##########
   ##############################
catch
   exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try

   ############################
   ######### throw exc ##########
   ############################
catch
   exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try

                    ###########
  ######### throw exc ##########
  ################################
catch
  exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try


   ######### throw exc ##########
   ##############################
catch
   exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try

          throw exc ##########
  ##############################
catch
  exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try

    throw exc ##########
  ##############################
catch
  exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
try

    throw exc ##########
  ##############################
catch
  exc → someValue
end
```

PRZYKŁAD: WYJĄTKI

```
someValue
```

DEMO: WYJĄTKI

PRZYKŁAD: STAŁA LOKALNO-GLOBALNA

```
handle
   ##################################
   ##################################
   ######### ask () #########
   ##################################
with
   ...
end
```

PRZYKŁAD: STAŁA LOKALNO-GLOBALNA

```
handle
   ##################################
   ##################################
   ######### ask () #########
   ##################################
with
   ...
end
```

PRZYKŁAD: STAŁA LOKALNO-GLOBALNA

```
handle
  ##############################
  ##############################
  ######### ask () #########
  ##############################
with
  ...
end
```

PRZYKŁAD: STAŁA LOKALNO-GLOBALNA

DEMO: STAŁA LOKALNO-GLOBALNA

PRZYKŁAD: NIEDETERMINIZM

```
handle
  ##### flip () ####
with
  ...
end
```

PRZYKŁAD: NIEDETERMINIZM

```
handle
  ##### flip () ####
with
  ...
end
```

PRZYKŁAD: NIEDETERMINIZM

```
[##### True ####; ##### False ####]
```

DEMO: NIEDETERMINIZM