MP25 @ II UWr 6 maja 2025 r.

Zadanie nr 2 na pracownię

Arkusz kalkulacyjny

Celem pracowni jest stworzenie programu, który wylicza wartość arkusza kalkulacyjnego. Wejściem takiego programu jest arkusz zawierający w komórkach wyrażenia w języku FUN rozbudowanym o dodatkową konstrukcję `{n,k}`, która ozancza wartość komórki w `n`-tym wierszu i `k`-tej kolumnie. Wyjściem jest arkusz, w którym zawartością komórek są wartości tych wyrażeń.

Przykład

Arkusz może być podany jako plik csv bez nagłówków:

```
$ cat examples/ex1.fs
"10","50"
"0 - {1,1}","{0,0} + {0,1}"
```

Wynikiem są obliczone wartości tych wyrażeń:

```
$ dune exec spreadsheet examples/ex1.fs
10,50
-60,60
```

W szczególności:

- Komórka o współrzędnych $0, 0$ zawiera wyrażenie `"10"`, którego wartością jest oczywiście `10`
- Komórka o współrzędnych $0, 1$ zawiera wyrażenie `"50"`, którego wartością jest oczywiście `50`
- Komórka o współrzędnych $1, 1$ zawiera wyrażenie `"{0,0} + {0,1}"`, którego wartością jest suma dwóch powyższych wyrażeń, czyli `60`
- Komórka o współrzędnych $0, 1$ zawiera wyrażenie `"0 - {1,1}"`, którego wartością jest odwrotność powyższego wyrażenia, czyli `-60`

Proszę zwrócić uwagę, że wartości komórek mogą od siebie zależeć, np. w powyższym przykładzie komórka $1,1$ zależy od komórek $0,0$ i $0,1$, a komórka $1,0$ zależy od $1,1$. Jeśli zależności te tworzą cykl, wyliczenie arkusza powinno zakończyć się błędem. Na przykład:

```
$ cat examples/ex3.fs
"{1,0} + 3"
"{2,0} * 10"
"{0,0} - 17"

$ dune exec spreadsheet examples/ex3.fs
Error: Circular dependency
```

Język

Językiem, którego wyrażenia mogą być wartościami wejściowymi komórek arkusza jest FUN rozbudowany o konstrukcję `{n,k}`. Składnia abstrakcyjna tej konstrukcji to:

```
type expr =
  ...
  | Cell of int * int
```

W składni konkretnej reprezentowana jest ona jako:

```
%token LBRACE
%token RBRACE

  ...

rule read =
    parse
    ...
    | "{" { LBRACE }
    | "}" { RBRACE }

  ...

base_expr:
  ...
  | LBRACE; x = INT; COMMA; y = INT; RBRACE { Cell(x, y) }
```

Zadanie

Na SKOS-ie znajdą Państwo szablon, który ma strukturę taką samą jak interpreter języka FUN z wykładu rozbudowany o możliwość wczytywania plików csv

z wejścia. Zadaniem jest napisanie fragmentu, który dokonuje samego obliczenia wartości arkusza. Wszystkie zmiany należy wprowadzić w pliku `interp.ml`. W szczególności:

- Należy zdefiniować funkcję

  ```
  eval_spreadsheet : expr list list -> value list list option
  ```

  której:

  - wejściem jest arkusz zawierający wyrażenia (w składni abstrakcyjnej) reprezentowany jako lista list (jest to lista wierszy, a każdy wiersz to lista wyrażeń w poszczególnych kolumnach),
  - a wyjściem jest arkusz (lista wierszy) zawierający wartości. Wyjście dodatkowo jest typu `option`, tak że w razie wykrycia cyklu zależności, wynikiem funkcji `eval_spreadsheet` powinno być `None`.

- Dodatkowo, należy zmodyfikować resztę pliku `interp.ml` według potrzeb. W szczególności, zamieszczony w szablonie tam interpreter nie obsługuje konstrukcji `Cell`.

Nie należy modyfikować innych plików niż `interp.ml`, a także zmieniać typu `value`.

Uwagi

- Zapis `{n,k}` w składni konkretnej odpowiada konstrukcji `Cell(n,k)` w składni abstrakcyjnej. Nieformalnie należy rozumieć takie wyrażenie jako wartość wyrażenia znajdującego się w komórce w `n`-tej kolumnie i `k`-tym wierszu, licząc od zera. Jest to więc zapis stosowany dla elementów macierzy, a NIE zapis zwykle używany w arkuszach kalkulacyjnych, gdzie nazwa komórki, np. `C17` oznacza kolumnę `C` i `17`-ty wiersz (więc najpierw kolumna).
- Proszę zwrócić uwagę, że konstruktor `Cell` jako swój argument przyjmuje parę intów, a nie parę wyrażeń. Dzięki temu można statycznie zbadać, od których komórek zależy wartość danych komórek. Wiedza ta może przydać się do rozwiązania, ale nie musi (bo jest kilka sposobów na rozwiązanie tego zadania).
- Można dowolnie zmieniać sam interpreter. W szablonie znajduje się interpreter języka FUN z wykładu, ale być może (w zależności od sposobu rozwiązania zadania) lepiej będzie Państwu pasował np. interpreter w stylu interpretera języka FUN_MONAD.

- W tym zadaniu wolno używać OCamlowego mutowalnego stanu i OCamlowych wyjątków (ale nie trzeba, w zależności od sposobu rozwiązania).

Dodatkowy przykład

```
$ cat examples/ex2.fs
"funrec fib n -> if n <= 1 then n else fib (n - 1) + fib (n - 2)","0"
"{0,0} 1","1"
"{0,0} 2","{0,1} + {1,1}"
"{0,0} 3","{1,1} + {2,1}"
"{0,0} 4","{2,1} + {3,1}"
"{0,0} 5","{3,1} + {4,1}"
"{0,0} 6","{4,1} + {5,1}"

$ dune exec spreadsheet examples/ex2.fs
<fun>,0
1,1
1,1
2,2
3,3
5,5
8,8
```