Летняя практика 2015: Type synonyms preservation problem

Денис Николаевич Москвин

СПбАУ РАН

10.06.2015

Type synonyms

- Синонимы типов позволяют скрывать внутреннюю структуру сложно устроенного типа:

```
type String = [Char]

type IO a = World -> (a, World)
```

- После объявления синонимы типов можно свободно использовать вместо исходного типа:

```
strConcat :: String -> String -> String
strConcat = (++)

withFile :: FilePath -> IOMode -> (Handle -> IO r) -> IO r
withFile = ...
```

Type synonyms and type inference

- При выводе типов необходимо решать уравнения на типы.
- При этом может потребоваться раскрыть определение синонима.
- Иногда этого не требуется, и синоним типа сохраняется:

Сессия GHCi

```
> :t strConcat "ab"
strConcat "ab" :: String -> String
> :t strConcat "ab" "cd"
strConcat "ab" "cd" :: String
> :t map (strConcat "ab") ["cd", "ef"]
map (strConcat "ab") ["cd", "ef"] :: [String]
```

Type synonym disappearance

- Когда раскрытие определения синонима требуется, то синоним может исчезнуть.
- Например, для сечения `(++) :: [a] -> [a] -> [a]` имеем

Сессия GHCi

```
> :t (++ strConcat "ab" "cd")
(++ strConcat "ab" "cd") :: [Char] -> [Char]
```

- Иногда синоним исчезает частично. Например, для `reverse :: [a] -> [a]`

Сессия GHCi

```
> :t \x -> reverse (strConcat "ab" x)
\x -> reverse (strConcat "ab" x) :: String -> [Char]
```

Type synonym disappearance (2)

- В более сложных случаях могут наблюдаться другие интересные артефакты алгоритма вывода типов.
- Например, зависимость от порядка:

Сессия GHCi

```
> :t [strConcat "ab" "cd", "ef" ++ "gh"]
[strConcat "ab" "cd", "ef" ++ "gh"] :: [String]
> :t ["ab" ++ "cd", strConcat "ef" "gh"]
["ab" ++ "cd", strConcat "ef" "gh"] :: [[Char]]
```

Type synonyms preservation problem

- Возникает задача: *обеспечить сохранение синонима типа при выводе типа*.
- Это требует некоторой модификации алгоритма вывода.
- Необходимо:
  - при построении системы уравнений на типы сохранять информацию о наличии синонима;
  - при унификации не терять эту информацию;
  - после вывода типа обеспечить восстановление синонима.
- **Базовый этап:** реализовать такой алгоритм для STT, расширенной синонимами типов. Постараться сделать это максимально эффективно, оценить асимптотику.
- **Продвинутый этап:** реализовать такой алгоритм для системы типов Haskell (напрямую, или итеративно, расширяя STT различными типовыми конструкциями, допустимыми в Haskell).