Functional logic programming

Giorgio Marinelli[1]

June 3, 2020

[1]`https://giorgiomarinelli.it/cv`

Introduction

Functional logic programming is the combination of two kind of declarative languages:

- Functional languages (eg. LISP, ML, Haskell, ...)
- Logic languages (eg. Prolog, Datalog, ...)

Introduction

An example of a functional logic programming language is Curry[2]

It is based on the Haskell language:

- Statically typed
- Purely functional
- Type inference
- Lazy evaluation

[2]`http://www.curry-language.org/`

Introduction

It has *functional logic programming* features, like:

- Narrowing
- Functional patterns
- Non-determinism
- Strategies

There exist two main implementation of the language:

- PAKCS: compile Curry programs into Prolog programs
- KiCS2: compile Curry programs into Haskell programs

Narrowing

One of the main feature in the Curry language is Narrowing: a combination of variable instantiation and term reduction (originally introduced in automated theorem proving).

```
last :: [a] -> a
last xs | zs ++ [e] =:= xs   =   e
  where zs, e free
```

Functional pattern

The `last` rule can be reformulated using *functional pattern*: a pattern with a function inside.

```
last :: [a] -> a
last (zs ++ [e]) = e
```

Haskell, do not allow this rule because it is not constructor based.

Non-determinism

Curry allow to define non deterministic computations (or operations).

```
(?) :: a -> a -> a
x ? y = x
x ? y = y
```

We can define an non deterministic operation like flipping a coin:

```
coin :: Int
coin = 0 ? 1
```

Demo