Curry Crash Course

Sebastian Fischer

Bonn 2013
Functional - Logic Programming

**functional**
- algebraic data types
- higher-order functions
- laziness, monadic IO

**logic**
- free variables for unknown values
- first-class nondeterminism and failure
- built-in search

Hasseil with simpler type system (no type classes, GADTs, ...)
Implementations

PTKCS
- based on Prolog
- many experimental features

KCC
- based on C
- fewer features

KiCS²
- based on Haskell
- different search strategies
Mountains

not below baseline

starts and ends on baseline

side 17
Mountains

1. Enumerate all mountains of size 7!

2. How many mountains exist of size 17?
Arithmetic Expressions

\[ 1 + 2 \quad 3 \times (5 + 4) \quad \frac{7}{2} - 3 \]

\[ [e_n] = n \quad \text{for integer } n \]

\[ [e_1 + e_2] = [e_1] + [e_2] \]

\[ [e_1 - e_2] = [e_1] - [e_2] \]

\[ [e_1 \times e_2] = [e_1] \times [e_2] \]

\[ [e_1 / e_2] = [e_1] / n \quad \text{if } [e_2] = n \neq 0 \]
Replacing Subexpressions

\[(1+2)[8] \langle 1 \rangle = 3 + 2\]

expression \[\downarrow\]

replacement \[\leftarrow\] position

\[((2+3)*7)[8/2] \langle 1,2 \rangle\]

= \[((2+8/2)*7)\]
Replacing Subexpressions

\[ e^{[\sigma]}_\langle \rangle = \sigma \]

\[
(e_1 \circ e_2)^{[\sigma]}_\langle 1, p \rangle = e_1^{[\sigma]}_p \circ e_2
\]

\[
(e_1 \circ e_2)^{[\sigma]}_\langle 2, p \rangle = e_1 \circ e_2^{[\sigma]}_p
\]

for \( o \in \{+1, -1, *, 1, /\} \)
Selecting Literals

\[
literal (\lambda + 2) \rightarrow 1
\]

\[
literal (\lambda + 2) \rightarrow 2
\]

\[
literal (\text{e}[n]p) \rightarrow n
\]

for integer \( n \)
\textbf{Simplification}

$1 + 0 \rightarrow 1 \quad 2 + 0 \rightarrow 2$

$\text{simples } (e_1[e_2+0]_p) = e_1[e_2]_p$

$4 \times 2 + 0 \rightarrow 42$
Simplification

more Complex \( e = e + O \)
more Complex \( e = O + e \)

\[
\vdots
\]

simpler \( (e_1 [\text{more Complex } e_2]_p) \)
\[ \rightarrow e_1 [e_2]_p \]
Simplification

simplify \( e = e \) if simpler fails

simplify \( e = \text{simplify} \left( \text{simpler} \ e \right) \) otherwise
Passes

Type `Passes a = String → (a, String)`

\((\text{P}11\text{I}) \equiv \text{Passes } a \rightarrow \text{Passes } a \rightarrow \text{Passes } a\)

*implicit nondeterminism*
1. Enumerating all strings of length at most 3 for expressions that evaluate to 4!

2. How many strings of length at most 5 correspond to expressions that evaluate to 4! 2?
Summary

Built-in failure simplifies programs that may fail. Evaluators, division by zero.

Built-in nondeterminism simplifies programs that perform search. Simplification, passing.

Programs can be run backwards to search arguments for results. Passes + evaluators.