Language description

Tomasz Kanas

April 12, 2019

1 General concept

Language will be pure functional, static typed with eager evaluation. Language will have recursive algebraic (but not polymorphic) types, with a bit unusual syntax: type unions will be indexed (instead of tagged). Language will also feature name overloading, which means that there can exist many values with the same names, as long as they have different types (functions need to have different argument types). Input/Output will be implemented in reactive style, which means that there will be Input stream which will behave as char value, except that the computation will be carried on every input character sequentially. To enhance reactive style, language will feature "stream" data types. Each stream will have input (arguments), state (private) and output (public), and will behave similarly as a function from (labeled) tuple to (labeled) tuple, except that every expression inside string will be implicitly function of input, state and output (from previous computation).

2 Language features

- Basic types: int, char, bool, void
- Basic (int and bool) arithmetic, comparisons, 'let ... in', 'if ... then ... else ... '
- Anonymous and named functions, recursion, partial application, higher-level functions
- Runtime error handling
- Algebraic, recursive types, type aliases
- Static name binding
- Static typing
- Multi-level pattern matching ('match ... with ... ')
- Syntax sugar for list types: constructors and patterns 'head:tail', '[items]'
- name overloading
- stream data types
- reactive Input/Output
- type reconstruction

Language complexity estimation

For full implementation of this language i hope to get 30 points.