

System F Language Specification

Syntax

Expressions

e	$:=$	lit eid (e) $e[\tau]$ $e_1 e_2$ $e_1 \text{ op } e_2$ $\lambda \text{ pat. } e$ $\Lambda \text{ tid. } e$ (e_1, \dots, e_n) $\text{let pat} = e_1 \text{ in } e_2$ $\text{if } e_1 \text{ then } e_2 \text{ else } e_3$	literals expression identifier parenthesized type concretization application binary operation lambda abstraction type abstraction n -tuples, $n \geq 2$ let binding if expression
lit	$:=$	null $\text{true} \mid \text{false}$ $\dots \mid \sim 2 \mid \sim 1 \mid 0 \mid 1 \mid 2 \mid \dots$	unit literal: Unit boolean literals: Bool 64-bit signed ints: Int
pat	$:=$	$_ : \tau$ $eid : \tau$ (pat_1, \dots, pat_n)	discarded variable type-annotated variable n -tuple destructor, $n \geq 2$

Types

τ	$:=$	tid (τ) $\tau_1 \rightarrow \tau_2$ $\forall \text{ tid. } \tau$ (τ_1, \dots, τ_n) $\text{Int} \mid \text{Bool} \mid \text{Unit}$	type identifier parenthesized arrow types universal types tuple types, $n \geq 2$ built-in types
--------	------	---	---

Declarations

δ	$:=$	$\text{let pat} = e$	declaration
----------	------	----------------------	-------------

Semantics:

Call-by-value big step semantics.

When a bound variable is bound again, the new binding takes over.

There is no one-type tuples

Lexical scope.