System F Language Specification

Syntax:

e	:=	lit	literals
		eid	expression identifier
		$e[\tau]$	type application
		(e)	parenthesized
		$e_1\ e_2$	application
		$(\overline{e}^{(,)\geq 2})$	tuples
		$e_1\ op\ e_2$	infix operation
		any $\overline{tid}^{(,)\geq 1}$**.** e	parametric polymorphism
		if e_1 **then** e_2 **else** e_3	if expression
		let eid **:** τ **=** e_1 **in** e_2	let binding
		lambda $\overline{eid\ \textbf{:}\ \tau}^{(,)\geq 1}$**.** e	anonymous function
		match e **with** $\overline{pat\ \texttt{=>}\ \mathrm{e}}^{(\vert)\geq 1}$	pattern destructing
τ	:=	tid	type identifier
		(τ)	parenthesized
		$\tau_1 * \tau_2$	tuple types
		τ_1 **->** τ_2	arrow types
		forall $\overline{tid}^{(,)\geq 1}$**.** τ	universal types
		Int | **Bool** | **Unit**	Built-in types
lit	:=	**null**	unit literal
		true | **false**	boolean literals
		...| **~2** | **~1** | **0** | **1** | **2** | ...	64 bit signed integers
pat	:=	eid **:** τ	Binds a single expression
		$(\overline{pat}^{(,)\geq 2})$	Destructs a tuple

Multiple argument **lambda**'s (and **forall**'s) are syntactic sugar for nested functions. For instance,

$$\textbf{lambda}\ \mathrm{x}\colon \textbf{int},\ \mathrm{y}\colon \textbf{int}.\ \mathrm{x} + \mathrm{y} \overset{\text{def}}{=} \textbf{lambda}\ \mathrm{x}\colon \textbf{int}.\ \textbf{lambda}\ \mathrm{y}\colon \textbf{int}.\ \mathrm{x} + \mathrm{y}$$

Semantics:

CBV big step semantics with capture-avoiding substitution.
When a bound variable is bound again, the new binding takes over.
There is no one-type tuples Lexical scope.