Essentials of Programming Languages

https://proglang.informatik.uni-freiburg.de/teaching/konzepte/2018ss/

Language 4 – Simply Typed Lambda Calculus

2018-04-18

Lambda calculus, with types!

The lambda calculus returns, but with types! We will consider a call-by-value small step lambda calculus with integers. Lambdas are annotated with the types of their argument.

Implement a small step call-by-value semantics, similar to the previous exercise sheet, and a type-checking judgement (using `define-judgment-form`). This judgement will have three input parameters (the environment, the expression and the type) and no output. You will need to implement the Martelli-Montanari unification algorithm.

e	::=	x	Variables
	|	$(e\ e\ \ldots)$	Application
	|	$\lambda(x:\tau).e$	Abstraction
	|	$number \mid + \mid \ldots$	Arithmetic operations
	|	fix	Fixpoint combinator
τ	::=	α	Type Variables
	|	$\tau \to \tau$	Function types
	|	`Int`	Integer types

Exercise 1 (Testing soundness)
Express preservation and progress as properties of your language. Implement random tests for them.

Exercise 2 (Tuples)
Add tuples to your language.

e	::=	$\ldots$	
	|	$(*\ e\ e\ \ldots)$	Tuple
	|	$(\text{proj}\ \ i\ e)$	Projection
τ	::=	$\ldots$	
	|	$(*\ \tau\ \tau \ldots)$	Tuple types

Exercise 3 (Sum types)
Add sum types to your language.

e	::=	$\ldots$	
	|	$(\text{label}\ \ e)$	Constructor
	|	$(\text{match}\ \ e\ (\text{label}\ \ x\ \ e) \ldots)$	Matching
τ	::=	$\ldots$	
	|	$((\text{label}\ \ \tau)\ \ldots)$	Sum types

Exercise 4 (Bonus: Records)
Provide a syntax and a semantics for records.

1 Type inference

We now consider a variant of the previous language were lambdas are *not* annotated with a type. Implement the simple type inference algorithm by John Mitchell presented in the lecture. First implement it on the simply typed lambda calculus. Can you extend it to tuples, sums and records?