Theory of Types and Programming Languages Fall 2022

Week 8

Plan

PREVIOUSLY: unit, sequencing, let, pairs, sums, recursion, state

TODAY:

1. mutable state (continued)
2. Curry-Howard isomorphism

NEXT: polymorphic (not so simple) typing
NEXT: dependent type systems

References and Mutable State

[See slide deck from last week.]

The Curry-Howard Correspondence

In *constructive logics*, a proof of P must provide *evidence* for P.

- "law of the excluded middle" — $P \vee \neg P$ — not recognized.

A proof of $P \wedge Q$ is a *pair* of evidence for P and evidence for Q.

A proof of $P \Rightarrow Q$ is a *procedure* for transforming evidence for P into evidence for Q.

Propositions as Types

Logic	Programming languages
propositions	types
proposition $P \Rightarrow Q$	type $\mathtt{P} \to \mathtt{Q}$
proposition $P \wedge Q$	type $\mathtt{P} \times \mathtt{Q}$
proof of proposition P	term $\mathtt{t}$ of type $\mathtt{P}$
proposition P is provable	type $\mathtt{P}$ is inhabited (by some term)
	evaluation

Propositions as Types

Logic	Programming languages
propositions	types
proposition $P \Rightarrow Q$	type $\mathtt{P} \to \mathtt{Q}$
proposition $P \wedge Q$	type $\mathtt{P} \times \mathtt{Q}$
proof of proposition P	term $\mathtt{t}$ of type $\mathtt{P}$
proposition P is provable	type $\mathtt{P}$ is inhabited (by some term)
proof simplification (a.k.a. "cut elimination")	evaluation