Computer Science 131
Programming Languages

November 16, 2000

Typed Lambda Calculi and Logic

Pure Simply-Typed λ-Calculus

- Syntax

M,N ::=	**x**		*variables*
	|	λ**x:t.M**	*functions*
	|	**M N**	*applications*
t,u ::=	a_1 **|** a_2 **|** ...		*type variables*
	|	**t -> u**	*function types*

One-step b-Reduction

- The relation $\rightarrow_b$ is defined by:

$$\frac{}{(\lambda x{:}t.M)N \rightarrow_b M[x \rightarrow N]}$$

$$\frac{M \rightarrow_b M'}{M\ N \rightarrow_b M'\ N} \qquad \frac{N \rightarrow_b N'}{M\ N \rightarrow_b M\ N'}$$

$$\frac{M \rightarrow_b M'}{\lambda x{:}t.M \rightarrow_b \lambda x{:}t.M'}$$

Static Semantics

$$\frac{}{\ldots, \mathbf{x:t}, \ldots \vdash \mathbf{x : t}}$$

$$\frac{\Gamma, \mathbf{x:t} \vdash \mathbf{M : u}}{\Gamma \vdash \mathbf{(\lambda x:t.M) : t \to u}}$$

$$\frac{\Gamma \vdash \mathbf{M : t \to u} \qquad \Gamma \vdash \mathbf{N : t}}{\Gamma \vdash \mathbf{M\ N : u}}$$

Encodings

- Given a fixed type t, we can define

$\underline{\mathbf{tt}}_t$:= λx:t.λy:t.x : t->t->t

$\underline{\mathbf{ff}}_t$:= λx:t.λy:t.y : t->t->t

$\mathbf{0}_t$:= λb:t.λf:(t->t).b

$\underline{\mathbf{1}}_t$:= λb:t.λf:(t->t).(f(b))

$\underline{\mathbf{n}}_t$:= λb:t.λf:(t->t).(f^n(b))

: t -> (t->t) -> t

Encodings

- Not all untyped lambda terms can be given types.
  - E.g., no type **t** makes **λx:t.xx** typecheck
  - The **Y** and Q combinators also can't be typed.

- Consequence: *pure simply-typed* λ-calculus is "less expressive".

Properties

1. Confluence holds for well-typed terms.

2. Convertibility is *consistent*.
   - There are terms which are not convertible.
   - Distinct normal forms are not convertible.

3. The β-reduction relation is *strongly normalizing*
   - No term permits an infinite sequence of reductions

4. Convertibility is *decidable.*

This slide intentionally left blank

Extension: Adding Pairs

- Syntax as in NQSML

```
M,N ::=  ...
       |  <M,N>          pairs
       |  M.1 | M.2      projections
t,u ::=  ...
       |  t * u          pair types
```

Typing and Reduction

$$\frac{\Gamma \vdash \mathtt{M} : \mathtt{t} \qquad \Gamma \vdash \mathtt{N} : \mathtt{u}}{\Gamma \vdash \mathtt{<M,N>} : \mathtt{t*u}}$$

$$\frac{\Gamma \vdash \mathtt{M} : \mathtt{t*u}}{\Gamma \vdash \mathtt{M.1} : \mathtt{t}} \qquad \frac{\Gamma \vdash \mathtt{M} : \mathtt{t*u}}{\Gamma \vdash \mathtt{M.2} : \mathtt{u}}$$

$$\frac{}{\mathtt{<M,N>.1} \rightarrow \mathtt{M}} \qquad \frac{}{\mathtt{<M,N>.2} \rightarrow \mathtt{N}}$$

Plus "congruence rules"

Extension: Sum Types

```
M,N ::=  ...
      |  inl_{t,u} M          tagging
      |  inr_{t,u} M
      |  (case M of           case analysis
            inl(x)=>N_1
          | inr(x)=>N_2)

t,u ::=  ...
      |  t + u                sum types
```

Typing and Reduction

$$\frac{\Gamma \vdash \mathbf{M} : \mathbf{t}}{\Gamma \vdash \mathbf{inl}_{\mathbf{t,u}}(\mathbf{M}) : \mathbf{t+u}} \qquad \frac{\Gamma \vdash \mathbf{M} : \mathbf{u}}{\Gamma \vdash \mathbf{inr}_{\mathbf{t,u}}(\mathbf{M}) : \mathbf{t+u}}$$

$$\frac{\Gamma \vdash \mathbf{M} : \mathbf{t_1+t_2} \qquad \Gamma, \mathbf{x}:\mathbf{t_1} \vdash \mathbf{N_1} : \mathbf{u} \qquad \Gamma, \mathbf{x}:\mathbf{t_2} \vdash \mathbf{N_2} : \mathbf{u}}{\Gamma \vdash \mathbf{(case\ M\ of\ inl(x)=>N_1|inr(x)=>N_2)} : \mathbf{u}}$$

$$\frac{}{\mathbf{(case\ inl_{t,u}(M)\ of\ inl(x)=>N_1 \mid inr(x)=>N_2)} \rightarrow \mathbf{N_1[x \rightarrow M]}}$$

Plus "$\mathbf{case\ inr_{t,u}(M)...}$*"*
and congruence rules

Typed Combinatory Logic

- Syntax

M,N	**::=**	$\mathbf{S_{t,u,v}}$	
	|	$\mathbf{K_{t,u}}$	
	|	**M N**	*applications*
t,u,v	**::=**	a_1 **|** a_2 **|** ...	*type variables*
	|	**t -> u**	*function types*

Typing Rules

$$\frac{}{\mathbf{K}_{t,u} : t \to u \to t}$$

$$\frac{}{\mathbf{S}_{t,u,v} : (t \to u \to v) \to (t \to u) \to (t \to v)}$$

$$\frac{M : t \to u \qquad N : t}{M\ N : u}$$

Claim: $\mathbf{S}_{t,t\to t,t}\ \mathbf{K}_{t,t\to t}\ \mathbf{K}_{t,t} : t \to t$

Recall: Static Semantics

$$\frac{}{\ldots, \mathbf{x:t}, \ldots \vdash \mathbf{x : t}}$$

$$\frac{\Gamma, \mathbf{x:t} \vdash \mathbf{M : u}}{\Gamma \vdash \mathbf{(\lambda x:t.M) : t \to u}} \qquad \frac{\Gamma \vdash \mathbf{M : t \to u} \quad \Gamma \vdash \mathbf{N : t}}{\Gamma \vdash \mathbf{M\ N : u}}$$

$$\frac{\Gamma \vdash \mathbf{M : t} \quad \Gamma \vdash \mathbf{N : u}}{\Gamma \vdash \mathbf{\langle M, N \rangle : t * u}}$$

$$\frac{\Gamma \vdash \mathbf{M : t * u}}{\Gamma \vdash \mathbf{M.1 : t}} \qquad \frac{\Gamma \vdash \mathbf{M : t * u}}{\Gamma \vdash \mathbf{M.2 : u}}$$

Erasing All but the Types

$$\frac{}{\ldots, \mathbf{t}, \ldots \vdash \mathbf{t}}$$

$$\frac{G, \mathbf{t} \vdash \mathbf{u}}{G \vdash \mathbf{t}\text{->}\mathbf{u}} \qquad \frac{G \vdash \mathbf{t}\text{->}\mathbf{u} \quad G \vdash \mathbf{t}}{G \vdash \mathbf{u}}$$

$$\frac{G \vdash \mathbf{t} \quad G \vdash \mathbf{u}}{G \vdash \mathbf{t}*\mathbf{u}}$$

$$\frac{G \vdash \mathbf{t}*\mathbf{u}}{G \vdash \mathbf{t}} \qquad \frac{G \vdash \mathbf{t}*\mathbf{u}}{G \vdash \mathbf{u}}$$

Rules for Propositional Logic

$$\frac{}{\ldots, \mathbf{p}, \ldots \vdash \mathbf{p}}$$

$$\frac{\Gamma, \mathbf{p} \vdash \mathbf{q}}{\Gamma \vdash \mathbf{p} \to \mathbf{q}} \qquad \frac{\Gamma \vdash \mathbf{p} \to \mathbf{q} \quad \Gamma \vdash \mathbf{p}}{\Gamma \vdash \mathbf{q}}$$

$$\frac{\Gamma \vdash \mathbf{p} \quad \Gamma \vdash \mathbf{q}}{\Gamma \vdash \mathbf{p} \wedge \mathbf{q}}$$

$$\frac{\Gamma \vdash \mathbf{p} \wedge \mathbf{q}}{\Gamma \vdash \mathbf{p}} \qquad \frac{\Gamma \vdash \mathbf{p} \wedge \mathbf{q}}{\Gamma \vdash \mathbf{q}}$$

Curry-Howard Isomorphism

- a.k.a. "Proofs as programs", "Propositions as types"
- Every type corresponds to a logical proposition
  - Type variables correspond to propositional variables
  - Function types correspond to implications
  - Pair types correspond to conjunctions
- A proposition is provable if and only if there is a term of the corresponding type
  - Such types are said to be *inhabited*.
  - Typed λ-terms are encodings of proofs.

Examples

1. Show that

   $$\vdash p \Rightarrow (p \wedge p)$$

   by finding a term of type

   ```
   a -> (a * a)
   ```

2. Show that

   $$\vdash (p \Rightarrow (q \Rightarrow r)) \Rightarrow ((p \wedge q) \Rightarrow r))$$

   by finding a term of type

   ```
   (a->(b->g)) -> ((a*b)->g)
   ```

Extensions

- The **`true`** proposition corresponds to a non-empty type
  - Typically, **`unit`**.
- The **`false`** proposition corresponds to an empty type.
  - Typically, **`void`**.
  - Encode **`¬`**p as (p Þ **`false).`**
- Second-order predicate calculus: polymorphic types
  - Second-order = quantifying over *propositions*.
  - E.g., "a**.** a ®**(**a ***** a**)** **vs.** "**p.** **p**®**(p**Ù **p)**
- Propositional logic: dependent types
- Modal logic: run-time code generation
- Linear logic: linear types

Intuitionism

- Generally, λ-calculi correspond to *constructive* or *intuitionistic* logics.
  - E.g., no terms of type

`"a. a + (a->void)`	*("p. p or ¬p)*
`"a. ((a->void)->void) -> a`	*("p. ¬¬p Þ p)*
`"a."b.((a->b)->a) -> a`	*(Pierce's Law)*

- However, there has been some work on extending the isomorphism to classical logics
  - Corresponds to calculi with **`callcc`**-like operators

Proof Normalization

- Reductions as proof "simplifications".

$$\dfrac{\dfrac{\Gamma \vdash \mathbf{M} : \mathbf{t} \qquad \Gamma \vdash \mathbf{N} : \mathbf{u}}{\Gamma \vdash \mathbf{\langle M,N \rangle} : \mathbf{t*u}}}{\Gamma \vdash \mathbf{\langle M,N \rangle.1} : \mathbf{t}} \qquad\qquad \Gamma \vdash \mathbf{M} : \mathbf{t}$$

$$\dfrac{\dfrac{\Gamma \vdash \mathbf{t} \qquad \Gamma \vdash \mathbf{u}}{\Gamma \vdash \mathbf{t \dot{\cup} u}}}{\Gamma \vdash \mathbf{t}} \qquad\qquad \Gamma \vdash \mathbf{t}$$

Summary

λ-Calculus

- Type
- Term (program)
- Reduction

Logic

- Proposition
- Proof
- Proof Normalization

Fixed Points

- Suppose we add in recursion (fixed points)
  - As in NQSML
  - No longer true that reduction always terminates
  - There is a term of *every* type
- Corresponds to an inconsistent logic
  - i.e., logic where every proposition is provable

Typed Combinatory Logic

$$\frac{}{K_{t,u} : t \to u \to t}$$

$$\frac{}{S_{t,u,v} : (t \to u \to v) \to (t \to u) \to (t \to v)}$$

$$\frac{M : t \to u \qquad N : t}{M\ N : u}$$

Hilbert System for Implication

Axiom Schema

$$\frac{}{p \to (q \to p)}$$

$$\frac{}{(p \to (q \to r)) \to ((p \to q) \to (p \to r))}$$

Modus Ponens

$$\frac{p \to q \quad p}{q}$$