

CS20a: Models (Nov 12, 2002)

- Alternative models
 - Primitive recursive functions
 - Partial recursive functions
 - Recursion theorem
 - Rice's theorem
- Lambda Calculus
 - Recursion theorem
 - Rice's theorem
- Arithmetic
 - Recursion theorem
 - Godel's incompleteness theorem (aka Rice's theorem)



Primitive recursion

Functions \mathbb{N} tuples $\rightarrow \mathbb{N}$ tuples

The following functions are primitive recursive:

- $s(x) = x + 1$ (the successor function)
- $z(x) = 0$ (the zero function)
- $\pi_i^m(x_1, \dots, x_m) = x_i$ (projection)
- For any $f : \mathbb{N}^m \rightarrow \mathbb{N}^m$, and $g_1, \dots, g_m : \mathbb{N}^k \rightarrow \mathbb{N}$, the composition $f(g_1(\vec{x}), \dots, g_m(\vec{x}))$ is primitive recursive



Primitive recursion (conventional form)

Given $h : \mathbb{N}^{n-1} \rightarrow \mathbb{N}^m$, and $g : \mathbb{N}^{n+m} \rightarrow \mathbb{N}^m$, define $f : \mathbb{N}^n \rightarrow \mathbb{N}^m$ as follows:

- $f(0, \vec{x}) = h(\vec{x})$
- $f(s(y), \vec{x}) = g(y, \vec{x}, f(y, \vec{x}))$



For-programs

- Variables x, y, z over \mathbb{N}
- Simple assignments:
 - $x \leftarrow y$
 - $x \leftarrow s(y)$
 - $x \leftarrow 0$
- Induction: if p, q are *for-programs*, so are:
 - $p; q$ (sequential composition)
 - **if** $x = y$ **then** p **else** q (conditional)
 - **for** y **do** p **done** (for-loop)



For-loop: for y do p done

- Execute p , y times
- Assignments to y in p do not affect the number of iterations
- **for** $i = 1$ **to** y **do** p **done** is equiv to

```

i ← 0;
for y do
  p;
  i ← s(i)
done

```



Partial recursive functions (Godel)

- To capture r.e. computations, we need more
- A partial recursive computation includes the prim-rec computations, plus unbounded minimization



Single-step evaluation

- A single-step reduction is just a substitution
- Called “beta-reduction”

$$(\lambda v. e_1) e_2 \rightarrow_{\beta} e_1[e_2/v]$$



Fixpoint combinator

$$\begin{aligned} Y &\equiv (\lambda x. x x) (\lambda y. y y) \\ Y f &\rightarrow_{\beta}^* f (Y f) \end{aligned}$$

$$\begin{aligned} Y \text{ fact} &\equiv Y (\lambda i. \lambda i. \text{if } i = 0 \text{ then } 1 \text{ else } i * (f(i - 1))) \\ &\rightarrow \lambda i. \text{if } i = 0 \text{ then } 1 \text{ else } i * (Y \text{ fact } (i - 1)) \end{aligned}$$



Fixpoint theorem

Theorem For any function f , there is a value x s.t.
 $f(x) = f(f(x))$

Proof

$$f(Y f) \rightarrow^* f(f(Y f))$$



Lambda-calculus: Rice's theorem

Theorem Every nontrivial property of the λ -calculus functions is undecidable. That is, there is no total λ function $p : e \rightarrow \{\lambda x.\lambda y.x, \lambda x.\lambda y.y\}$ s.t.

- $p(e_k) \neq p(e_l)$ for some e_k, e_l
- If $e_1 = e_2$ then $p(e_1) = p(e_2)$

Proof If p exists, define

$$f(e) = \begin{cases} e_k & \text{if } p(e) = p(e_l) \\ e_l & \text{if } p(e) = p(e_k) \end{cases}$$

Then f has no fixpoint, contradiction.



Lambda-calculus: indexing

Define an indexing function $\ulcorner e \urcorner$ on lambda terms. Define as a sequence $i_1 :: i_2 :: \dots :: i_n$ that is the number of the term.

- $\ulcorner v_i \urcorner = [0; i]$
- $\ulcorner e_1 e_2 \urcorner = 1 :: \ulcorner e_1 \urcorner @ \ulcorner e_2 \urcorner$
- $\ulcorner \lambda v_i. e \urcorner = 2 :: i :: \ulcorner e \urcorner$

Define $\lambda_0, \lambda_1, \lambda_2, \dots$



Indexed Rice's theorem

Theorem Every nontrivial property of the λ -calculus functions is undecidable. That is, there is no total function $p : \mathbb{N} \rightarrow \{0, 1\}$ s.t.

- $p(k) \neq p(l)$ for some λ_k, λ_l
- If $\lambda_i = \lambda_j$ then $p(i) = p(j)$

Proof If p exists, define

$$f(i) = \begin{cases} k & \text{if } p(i) = p(l) \\ l & \text{if } p(i) = p(k) \end{cases}$$

Then f has no fixpoint, contradiction.



Arithmetic

- Let's do the same for arithmetic
- Arithmetic is:
 - Operators: $+$, $-$, $*$, $/$, \dots
 - First-order logic
- First, let's define logic



Defining logics

- A logic is defined in three parts
 - Syntax
 - Define what is "true"
 - Define derivation procedures



Defining the syntax

Start with a countable set of propositional letters P, Q, R, \dots
Define the propositions inductively:

- \top (true) is a proposition
- \perp (false) is a proposition
- Any propositional letter is a proposition
- If A is a proposition, so is $\neg A$ (negation)
- If A and B are propositions, so are
 - $A \wedge B$ (conjunction)
 - $A \vee B$ (disjunction)
 - $A \Rightarrow B$ (implication)



Sequents

- A *sequent* has the form $\Gamma \vdash \Delta$
- Δ is a list of propositions $\alpha_1, \dots, \alpha_n$
- Γ is a *context* containing a list of propositions β_1, \dots, β_n
- We can extend valuations to sequents, to get the following semantics:
 - A sequent $\beta_1, \dots, \beta_n \vdash \alpha_1, \dots, \alpha_n$ is true if some α_i is true whenever β_1, \dots, β_n are all true.



Derivations

- There are two kinds of inference rules
 - *Introduction* rules operate on the right of the turnstile
 - *Elimination* rules operate on the left of the turnstile
- The base axiom

$$\frac{}{\Gamma_1, \alpha, \Gamma_2 \vdash \Delta_1, \alpha, \Delta_2} \text{ axiom}$$



Introduction rules, part I

$$\frac{}{\Gamma \vdash \Delta_1, \top, \Delta_2} \text{ true intro}$$

$$\frac{\Gamma \vdash \Delta_1, \alpha, \Delta_2 \quad \Gamma \vdash \Delta_1, \beta, \Delta_2}{\Gamma \vdash \Delta_1, \alpha \wedge \beta, \Delta_2} \text{ and intro}$$

$$\frac{\Gamma \vdash \Delta_1, \alpha, \beta, \Delta_2}{\Gamma \vdash \Delta_1, \alpha \vee \beta, \Delta_2} \text{ or intro}$$



Introduction rules, part II

$$\frac{\Gamma, \alpha \vdash \Delta_1, \beta, \Delta_2}{\Gamma \vdash \Delta_1, \alpha \Rightarrow \beta, \Delta_2} \text{ implies intro}$$

$$\frac{\Gamma, \alpha \vdash \Delta_1, \Delta_2}{\Gamma \vdash \Delta_1, \neg \alpha, \Delta_2} \text{ not intro}$$



Elimination rules

$$\overline{\Gamma_1, \perp, \Gamma_2 \vdash \Delta} \text{ false elim}$$

$$\frac{\Gamma_1, \alpha, \beta, \Gamma_2 \vdash \Delta}{\Gamma_1, \alpha \wedge \beta, \Gamma_2 \vdash \Delta} \text{ and elim}$$

$$\frac{\Gamma_1, \alpha, \Gamma_2 \vdash \Delta \quad \Gamma_1, \beta, \Gamma_2 \vdash \Delta}{\Gamma_1, \alpha \vee \beta, \Gamma_2 \vdash \Delta} \text{ or elim}$$

$$\frac{\Gamma_1, \beta, \Gamma_2 \vdash \Delta \quad \Gamma_1, \Gamma_2 \vdash \alpha, \Delta}{\Gamma_1, \alpha \Rightarrow \beta, \Gamma_2 \vdash \Delta} \text{ implies elim}$$



Rule table

$\overline{\Gamma \vdash \Delta_1, \top, \Delta_2}$	$\overline{\Gamma_1, \perp, \Gamma_2 \vdash \Delta}$
$\frac{\Gamma \vdash \Delta_1, \alpha, \Delta_2 \quad \Gamma \vdash \Delta_1, \beta, \Delta_2}{\Gamma \vdash \Delta_1, \alpha \wedge \beta, \Delta_2}$	$\frac{\Gamma_1, \alpha, \beta, \Gamma_2 \vdash \Delta}{\Gamma_1, \alpha \wedge \beta, \Gamma_2 \vdash \Delta}$
$\frac{\Gamma \vdash \Delta_1, \alpha, \beta, \Delta_2}{\Gamma \vdash \Delta_1, \alpha \vee \beta, \Delta_2}$	$\frac{\Gamma_1, \alpha, \Gamma_2 \vdash \Delta \quad \Gamma_1, \beta, \Gamma_2 \vdash \Delta}{\Gamma_1, \alpha \vee \beta, \Gamma_2 \vdash \Delta}$
$\frac{\Gamma, \alpha \vdash \Delta_1, \beta, \Delta_2}{\Gamma \vdash \Delta_1, \alpha \Rightarrow \beta, \Delta_2}$	$\frac{\Gamma_1, \beta, \Gamma_2 \vdash \Delta \quad \Gamma_1, \Gamma_2 \vdash \alpha, \Delta}{\Gamma_1, \alpha \Rightarrow \beta, \Gamma_2 \vdash \Delta}$



Proving the law of excluded middle

- Every proposition is either true or false

$$\frac{\frac{\frac{\overline{A \vdash A}}{\vdash A, \neg A} 2}{\vdash A \vee \neg A} 1}{} 3$$



Pierce's law

$$\frac{\frac{\frac{\overline{A \vdash B, A}}{\vdash A \Rightarrow B, A} 3}{(A \Rightarrow B) \Rightarrow A \vdash A} 2}{\vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A} 1}{} 4$$



Currying

$$\frac{\frac{\frac{\overline{A, B \vdash A} 5}{A, B \vdash A \wedge B} 4}{(A \wedge B) \Rightarrow C, A, B \vdash C} 3}{(A \wedge B) \Rightarrow C \vdash A \Rightarrow B \Rightarrow C} 2}{\vdash ((A \wedge B) \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C))} 1}{} 6$$



A quantifier DeMorgan law

$$\frac{\frac{\frac{\frac{\frac{\frac{\varphi(c) \vdash \varphi(c)}{6}}{\forall x.\varphi(x) \vdash \varphi(c)}{5}}{\neg\varphi(c), \forall x.\varphi(x) \vdash}{4}}{\neg\varphi(c) \vdash \neg(\forall x.\varphi(x))}{3}}{\exists x.\neg\varphi(x) \vdash \neg(\forall x.\varphi(x))}{2}}{\vdash (\exists x.\neg\varphi(x)) \Rightarrow \neg(\forall x.\varphi(x))}{1}$$



Another proof

$$\frac{\frac{\frac{\frac{\frac{\varphi(c) \vdash \varphi(c)}{5}}{\varphi(c) \vdash \exists x.\varphi(x)}{5}}{(\exists x.\varphi(x)) \Rightarrow \psi, \varphi(c) \vdash \psi}{4}}{(\exists x.\varphi(x)) \Rightarrow \psi, \varphi(c) \Rightarrow \psi}{3}}{(\exists x.\varphi(x)) \Rightarrow \psi \vdash \forall x.\varphi(x) \Rightarrow \psi}{2}}{\vdash ((\exists x.\varphi(x)) \Rightarrow \psi) \Rightarrow (\forall x.\varphi(x) \Rightarrow \psi))}{1}$$



Formalizing arithmetic

The language of arithmetic:

$$\begin{aligned} e & ::= i \mid v \mid e + e \mid \dots \mid e/e \\ P & ::= e = e \mid e < e \mid \dots \\ & \mid \neg P \\ & \mid P \wedge P \mid P \vee P \mid P \Rightarrow P \\ & \mid \exists v.P[v] \mid \forall v.P[v] \end{aligned}$$

Example:

$$\forall x_1.\exists x_2.\forall x_3.\exists x_4.(x_3 + x_2 \leq (1 + x_1) * x_4)$$


