

# 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*  $f(g_1(\bar{x}), \dots, g_m(\bar{x}))$  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:

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



# *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

```
 $i \leftarrow 0;$   
for  $y$  do  
     $p;$   
     $i \leftarrow s(i)$   
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



# Unbounded minimization

- Find the least  $y$  s.t.  $g(\bar{x}, y) = 0$

$$f(\bar{x}) = \mu y. g(\bar{x}, y) = 0$$

where  $g$  is primitive recursive

- Alternatively:

$$f(\bar{x}) = \mu y. (g(\bar{x}, y) = 0 \wedge g(\bar{x}, z) \text{ is defined for } z \leq y)$$

where  $g$  does not have to be primitive recursive



# *While-programs*

The *while* programs are defined inductively:

- All for-programs are while-programs
- Add **while**  $x \neq y$  **do**  $p$  **done**, where  $p$  is a while-program



# *Indexing partial recursive funs*

$$\varphi_0, \varphi_1, \dots, \varphi_i, \dots$$

- $i$  is called the *index*
- $\varphi_i$  is the function, a semantic concept

Axiom of extensionality: two functions are equal, if they are equal on all arguments.

$$(\forall x. \varphi_i(x) = \varphi_j(x)) \Rightarrow \varphi_i = \varphi_j$$



# Gödel numbering

Gödel numbering:

- There is a universal function  $U(\langle i, \mathbf{x} \rangle) = \varphi_i(\mathbf{x})$
- For any  $m, n$  there is a total recursive function  $S_n^m$  (a partial recursive function defined on all inputs), s.t. for any  $i$  and  $x_1, \dots, x_n, y_1, \dots, y_m$

$$\varphi_{S_n^m(i, x_1, \dots, x_n)}(y_1, \dots, y_m) = \varphi_i(x_1, \dots, x_n, y_1, \dots, y_m)$$

An indexing is *acceptable* iff it satisfies these two properties.



# Recursion theorem

## Theorem (Fixed-point theorem)

For all total recursive functions  $f : \mathbb{N} \rightarrow \mathbb{N}$ , there is an index  $i$  s.t.  $\varphi_i = \varphi_{f(i)}$ .

## Proof

Take any  $v \in \mathbb{N}$ , and consider a recursive function that does the following on  $x$ :

1. Compute  $\varphi_v(v)$
2. If it halts, apply  $f$  to get  $f(\varphi_v(v))$
3. Use it as an index: compute  $\varphi_{f(\varphi_v(v))}(x)$



## Recursion theorem proof (cont.)

- There is a total recursive function  $h : \mathbb{N} \rightarrow \mathbb{N}$  s.t.  
 $\varphi_{h(v)}(x) = \varphi_{f(\varphi_v(v))}(x)$
- Since  $h$  is total, it has an index  $h = \varphi_u$

$$\varphi_{h(u)} = \varphi_{\varphi_u(u)} = \varphi_{f(\varphi_u(u))}$$

- So  $\varphi_u(u)$  is a fixed-point of  $f$ .



# Corollary: no effective minimal indices

## Theorem

There is no total recursive function  $s$  such that:

- $\forall i. \varphi_{s(i)} = \varphi_i$
- $\forall j. \varphi_{s(i)} = \varphi_j \Rightarrow j \geq s(i)$

**Proof** Suppose  $s$  exists, then define

$$f(i) = \begin{cases} s(i) - 1 & \text{if } s(i) > 0 \\ k & \text{if } s(i) = 0 \text{ where } \varphi_k \neq \varphi_0 \end{cases}$$

$f$  has no fixpoint, which is a contradiction.



# Rice's theorem

## Theorem (Rice's theorem)

Every non-trivial property of the partial recursive functions is undecidable. That is, there is no total recursive function  $p : \mathbb{N} \rightarrow \{0, 1\}$  s.t.

- $p(k) \neq p(l)$
- For any  $i, j$ , if  $\varphi_i = \varphi_j$ , then  $p(i) = p(j)$

**Proof** Suppose  $p$  exists. Define

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

$f$  is total, and it has no fixpoint.



# Effective padding

**Theorem** There is a total recursive function  $s$  such that:

$$\forall i. s(i) > i \wedge \varphi_i = \varphi_{s(i)}$$

**Proof** Given  $i$ , construct function  $f_0$  s.t.

$$f_0(x) = \begin{cases} i & \text{if } x \neq i \\ i + 1 & \text{if } x = i \end{cases}$$



# Effective padding

Now get a fixpoint  $\varphi_j = \varphi_{f_0(j)}$

- If  $j > i$ , let  $s(i) = j$
- If  $j = i$ , let  $s(i) = j + 1$
- If  $j < i$  then  $\varphi_j = \varphi_i$

– Redefine  $f_1(k) = \begin{cases} f_0(k) & \text{if } k \neq j \\ i + 1 & \text{if } k = j \end{cases}$

– Repeat



# *Lambda calculus*

- We have seen 3 models of computation
  - *Turing machines*
  - *Partial recursive functions*
  - *Lambda calculus*
- Lambda calculus
  - *All the usual properties hold (Rice's theorem)*



# *Lambda calculus*

- A starting point for reasoning about functions
- The foundation of most functional programming languages, including the Lisp and ML languages

$v$  ranges over a countable number of variables

$e$	$::=$	$v$	(variables)
		$e_1 e_2$	(function application)
		$\lambda v.e$	(function abstraction)



# 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

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

*Y fact*

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



# 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  $\lambda$ -calculus functions is undecidable. That is, there is no total  $\lambda$  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  $\lambda$ -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  $\lambda_k, \lambda_l$
- 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:  $+$ ,  $-$ ,  $*$ ,  $/$ , ...*
  - *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)



# Standard syntax definition

Propositions:

$$\begin{array}{l} e ::= \top \mid \perp \\ \mid P, Q, R, \dots \\ \mid \neg e \\ \mid e \wedge e \\ \mid e \vee e \\ \mid e \Rightarrow e \end{array}$$



# Semantics

- The semantics of constants:  $\top = 1, \perp = 0$
- The semantics of a propositional letter is its truth value.
- The semantics of a compound proposition is determined by truth tables.

$A$	$B$	$A \vee B$
0	0	0
0	1	1
1	0	1
1	1	1

$A$	$B$	$A \wedge B$
0	0	0
0	1	0
1	0	0
1	1	1

$A$	$B$	$A \wedge B$
0	0	1
0	1	1
1	0	0
1	1	1



# Extending the truth assignment

$A$	$B$	$C$	$A \wedge B$	$(A \wedge B) \Rightarrow C$
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	1



## Some definitions

- A *truth assignment*  $\mathcal{A}$  assigns a value to each propositional letter  $A$  a unique truth value  $\mathcal{A}(A) \in \{0, 1\}$
- A *truth valuation*  $\mathcal{V}$  assigns a truth value to each proposition (it can be constructed from a  $\mathcal{A}$  by following the truth tables).
- A proposition  $\alpha$  is *satisfiable* if there is a truth valuation  $\mathcal{V}(\alpha) = 1$
- A proposition  $\alpha$  is *true* (a tautology) if it is true in all valuations.



# A proof (Pierce's Law)

$A$	$B$	$A \Rightarrow B$	$(A \Rightarrow B) \Rightarrow A$	$((A \Rightarrow B) \Rightarrow A) \Rightarrow A$
0	0	1	0	1
0	1	1	0	1
1	0	0	1	1
1	1	1	1	1



# *Derivations*

- Truth tables are hard to use
- We want a mechanical method for proving propositions
- Use sequents and truth judgments



# Sequents

- A *sequent* has the form  $\Gamma \vdash \Delta$
- $\Delta$  is a list of propositions  $\alpha_1, \dots, \alpha_n$
- $\Gamma$  is a *context* containing a list of propositions  $\beta_1, \dots, \beta_n$
- 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  $\alpha_i$  is true whenever  $\beta_1, \dots, \beta_n$  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

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

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



# Elimination rules

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

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

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

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



# 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

$$\frac{\frac{\overline{A \vdash A}}{\vdash A, \neg A}}{\vdash A \vee \neg A}$$

3  
2  
1



# Pierce's law

$$\frac{\frac{\frac{\frac{\frac{}{A \vdash B, A}{}^4}{\vdash A \Rightarrow B, A}{}^3}{(A \Rightarrow B) \Rightarrow A \vdash A}{}^2}{\vdash ((A \Rightarrow B) \Rightarrow A) \Rightarrow A}{}^1}{A \vdash A}{}^5$$



# Currying

$$\begin{array}{c}
 \frac{}{A, B \vdash A} \quad 5 \quad \frac{}{A, B \vdash B} \quad 6 \\
 \hline
 \frac{}{A, B \vdash A \wedge B} \quad 4 \quad \frac{}{C, A, B \vdash C} \quad 6 \\
 \hline
 \frac{}{(A \wedge B) \Rightarrow C, A, B \vdash C} \quad 3 \\
 \hline
 \frac{}{(A \wedge B) \Rightarrow C \vdash A \Rightarrow B \Rightarrow C} \quad 2 \\
 \hline
 \frac{}{\vdash ((A \wedge B) \Rightarrow C) \Rightarrow (A \Rightarrow (B \Rightarrow C))} \quad 1
 \end{array}$$



# First-order logic

- Add *atomic formulas*  $f(a_1, \dots, a_n)$  of various arities.
- Add *atomic predicates*  $P(a_1, \dots, a_m)$  of various arities.

$$\begin{array}{l} a ::= f(a_1, \dots, a_n) \\ e ::= \top \mid \perp \\ \quad | P(a_1, \dots, a_m) \\ \quad | \neg e \\ \quad | e \wedge e \\ \quad | e \vee e \\ \quad | e \Rightarrow e \\ \quad | \forall v. e \\ \quad | \exists v. e \end{array}$$



## Some examples

- $\forall x. \text{big}(x) \Rightarrow \text{heavy}(x)$
- $\forall i. (i + 1) > i$
- $\forall i. \exists j. j > i$
- $\exists i. \forall j. j \leq i$



# Adding rules for FOL

$$\frac{\Gamma \vdash \Delta_1, P(t), \Delta_2}{\Gamma \vdash \Delta_1, \exists v.P(v), \Delta_2} \text{ exists intro}$$

$$\frac{\Gamma \vdash \Delta_1, P(c), \Delta_2 \text{ (new } c\text{)}}{\Gamma \vdash \Delta_1, \forall v.P(v), \Delta_2} \text{ all intro}$$

$$\frac{\Gamma_1, P(c), \Gamma_2 \vdash \Delta \text{ (new } c\text{)}}{\Gamma_1, \exists v.P(v), \Gamma_2 \vdash \Delta} \text{ exists elim}$$

$$\frac{\Gamma_1, \forall v.P(v), \Gamma_2, P(t) \vdash \Delta}{\Gamma_1, \forall v.P(v), \Gamma_2 \vdash \Delta} \text{ all elim}$$



# A quantifier DeMorgan law

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



# Another proof

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



# 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:

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

