TURING'S "ORACLE": FROM ABSOLUTE TO RELATIVE COMPUTABILITY--AND BACK

Solomon Feferman

Logic Seminar, Stanford, April 10, 2012

Plan

1. “Absolute” computability: machines and recursion theory.
2. Relative computability: degrees of unsolvability
3. Uniform relative computability: partial recursive functionals
4. Computability/recursion theory generalized to arbitrary structures
5. Significance of notions of relative computability for actual computation

1. “Absolute” effective computability
Origins

- Explication of the concept of effective computability (1933-1937)
- Church, Herbrand-Gödel, Turing, Post, Kleene
- Turing machines (1936-1937)
- Equivalence of the definitions
- The Church-Turing Thesis
- Register machines (Shepherdson, Sturgis, 1963)

‘Theory of Computation’ or ‘Recursion Theory’?

- Theory of computation emphasizes rule directed processes
- Recursion theory emphasizes a principal form of rule
- Ironically, Theoretical Computer Science is more concerned with the rules than the processes
- Soare’s campaign (e.g., ‘c.e.’ instead of ‘r.e.’, etc.)

Primitive Recursive Definition (Dedekind, Skolem)

- N = the natural numbers, $n' = n+1 = sc(n)$
- Defining effectively computable $f: N^k \to N$ by recursion equations.
- Primitive recursion: Explicit definition from 0, sc and previous functions, and
- for $k \geq 0$ and given g, h, and for $\underline{y} = (y_1,\ldots,y_k)$, $f(0, \underline{y}) = g(\underline{y})$, $\quad f(x', \underline{y}) = h(x, \underline{y}, f(x, \underline{y}))$

General Recursive Definition (Herbrand-Gödel)

- E a finite system of equations in f and auxiliary function symbols
- $E \vdash s = t$ if $(s = t)$ is derivable using substitution of numerals n^* for variables, and equals for equals.
- E computes f (say for $f: N \to N$) if
  $f(n) = m$ iff $E \vdash f(n^*) = m^*$
- f is general recursive if it is computable by some finite system of equations E.

General Recursive and Partial Recursive Functions

- <u>Theorem</u> The general recursive functions are the same as the Turing computable functions.
- Partial computable and partial recursive functions $f : N^k \to_p N$ (in the following, typically for k = 1)
- $f(n)\downarrow$, $f(n) \simeq m$
- E computes partial recursive f if whenever $E \vdash f(n^*) = m^*$ and $E \vdash f(n^*) = p^*$ then $m = p$.

Enumeration of Partial Rec. Fns.

- Kleene's Normal Form Theorem Each partial recursive $f : N \to_p N$ is representable in the form $f(x) \simeq U(\mu y. T(e, x, y))$ for some $e \in N$, where U, T are primitive recursive, $\mu y(\ldots) = \min y(\ldots)$.

- Enumeration Theorem The function $\{z\}(x) \simeq U(\mu y. T(z, x, y)$ is partial rec. and enumerates all unary partial rec. fns. for z = 0, 1, 2,... (~Universal Turing machine)

- The Halting Problems
  $H = \{(z,x) : \{z\}(x)\downarrow\}$, $K = \{x : \{x\}(x)\downarrow\}$

Decision Problems for A ⊆ N

- A is recursive (or decidable) if its characteristic fn. c_A is recursive
- The decision problem for A is effectively unsolvable if A is not recursive

Some Effectively Unsolvable Problems

- H
- K
- The Entscheidungsproblem for 1st order predicate logic
- Hilbert's 10th problem (Diophantine equations)
- The Word Problem for groups

Many-One Reduction and R.E. Sets

- $A \leq_m B$ iff for some general rec. f, $\forall x[x \in A \Leftrightarrow f(x) \in B]$
- If $A \leq_m B$ and A is not recursive then B is not recursive
- A is recursively enumerable (r.e.) if A is $\varnothing$ or the range of some (prim.) rec. f
- If B is r.e. and $A \leq_m B$ then A is r.e.

R. E. Sets (cont'd)

- The r.e. sets A are just those definable in the form $\forall x[x \in A \Leftrightarrow \exists y\, R(x, y)$ where R is (prim.) rec
- The unsolvable prob's above (H, K, etc.) are all r.e.
- If T is an effectively presented formal system then the set of Gödel nrs. of theorems of T is r.e.
- Every recursive set is r.e.
- <u>Fact</u>: If A is an r.e. set then $A \leq_m K$
- $\{z : \{z\}$ is total$\}$ is <u>not</u> r.e. $(\forall x \exists y T(z, x, y))$

2. Relative Effective Computability

- 'Oracle' computability (Turing 1939). A is effectively computable from B if it is computable by a machine which may call on an "oracle" for B.
- Write $f \leq g$ if f is computable from an oracle for g, and $A \leq B$ if $c_A \leq c_B$
- Can define $f \leq g$ iff for system of eqns. E
  $f(n) = m \Leftrightarrow E \cup Diag(g) \vdash f(n^*) = m^*$, where

  $Diag(g)$ is the set of all true $g(j^*) = k^*$.

Degrees of Unsolvability

- Post (1944): Define $A \equiv B \Leftrightarrow A \leq B \;\&\; B \leq A$,
- $\deg(A) = \{B : A \equiv B\}$, $\deg(A) \leq \deg(B)$ iff $A \leq B$
- $\underline{0} = \deg(N)$, $\underline{0}' = \deg(K)$
- <u>Fact</u>: If A is r.e. then $\deg(A) \leq \underline{0}'$

Post's Problem and Degree Theory

- Post's Problem Do there exist r.e. A with $\underline{0} < \deg(A) < \underline{0}'$?
- Yes! (Friedberg and Muchnik, independently, 1956) Construct A, B r.e. of incomparable degrees
- The priority method
- Structures of degrees of r.e. sets and degrees of arbitrary sets are both very complicated.

3. Uniform Relative Computability over N

- Define $f \leq g$ (via e) if f is computed from $E \cup \text{Diag}(g)$ where $e = \#(E)$.
- In degree theory f, g are given (or sought for) and ask whether there exists e s.t. $f \leq g$ (via e)
- Alternatively, fix e and define f as a uniform (partial) recursive function of g for all $g: N \rightarrow N$ via e; in general f is partial even for g total.

Partial Recursive Functionals

- Defn. A finite system of equations E determines a partial recursive functional $f = F(g)$ if for all partial g and n, m, p, if $E \cup Diag(g) \vdash f(n^*) = m^*, f(n^*) = p^*$ then $m = p$.
- Also write $F(g, n)$ for $(F(g))(n)$
- Lemma. If F is a partial rec. functional then it is (i) monotonic ($g \subseteq h \Rightarrow F(g) \subseteq F(h)$), (ii) continuous ($F(g,n) = m \Rightarrow F(h, n) = m$ for some finite $h \subseteq g$), and (iii) effective (g partial rec. $\Rightarrow$ F(g) partial rec.)

The Recursion Theorems

- <u>First Recursion Theorem</u> (Kleene 1952).
  For each partial rec. functional F there is a least solution to the equation
  $f = F(f)$, i.e. $f(x) \simeq F(f, x)$ for all x.
  Moreover the least fixed point (LFP) f is partial recursive.

- <u>Second Recursion Theorem</u> (Kleene 1938). For each partial rec. f we can find an index e such that $\{e\}(x) \simeq f(e, x)$ for all x.

Recursive Functionals of Finite Type over N

- Primitive rec. functionals of finite type over N (Gödel 1958)
- Partial rec. functionals of finite type over N (Kleene 1959)
- Theorem (Recursion in quantifiers, Kleene 1959). Let $\underline{E}(g) = 0$ iff $\exists n(g(n) = 0)$, else 1. Then f is partial rec. in $\underline{E}$ [$f \leq \underline{E}$] iff f is hyperarithmetic.

4. Generalized Recursion Theory (g.r.t.)

(a) Recursion over all ordinals (Takeuti 1960)

(b) Recursion over admissible ordinals and admissible sets (Kripke, Platek, 1964). The least admissible ordinal is ω; the least admissible ordinal > ω is the least non-recursive ordinal ("Church-Kleene" ω_1).

(c) _Degree theory on admissible ordinals (Sacks, Simpson, et al--generalization of the priority method)

Generalized Rec. Theory (cont'd)

- Computability/Recursion Theory over arbitrary structures (many workers from 1961 on).
- Turing machines and register machines on arbitrary structures (Friedman 1971).
- Partial rec. functionals of finite type on arbitrary structures (Platek 1966).
- Type two LFP schemata, uniform over structures (Moschovakis 1984, 1989).
- "While" schemata (Tucker and Zucker 2000).

5. Significance of Notions of Relative Computability for Actual Computation

- Computational practice and the theory of computation
- Turing machines are not a good model of actual computers (desktop or mainframe)
- Register machines are a better model (RAMs)
- Church-Turing thesis is accepted in principle by computer scientists, without effect on practice

Computational Theory and Practice

- Notions of absolute effective computability have little significance for practice
- Claim: The notions, but not the results, of relative computability, have much greater significance for practice
- Reasons: The requirements of efficiency, reliability, versatility and user-friendliness demand a modular organization of hardware and software.

Examples

- Built in functions and black boxes, for example for Boolean, arithmetical and analytic functions. Programs for an f from such g give $f \leq g$, but programmer doesn't need to know how box for g works.
- Functional programming languages, e.g. Lisp, ML, Scheme, Miranda, Haskell, etc. Moreover, flowchart diagrams are implicitly functional.

Examples (cont'd)

- Abstract data types (ADTs), e.g. integers, booleans, reals, lists, arrays, trees, etc. ADTs are structures considered up to isomorphism, independent of representation.
- "Hypercomputation": Online and Interactive Computation (cf. Soare 2009, and Nayebi presentation to come).

Coda: What has degree theory done for the theory of computation?

- On the face of it, complexity theory is a form of degree theory
- P, NP, co-NP, Exp, etc. complexity classes, space, time forms
- Many open separation problems: P =(?)NP, etc.
- It has been observed that recursion theoretic results generally relativize to any oracle.
- But relativized P = NP can go both ways (Baker, Gill, Solovay 1975).

Selected References

- J. Barwise (1975) *Admissible Sets and Structures*
- M. Davis (ed.)(1965), *The Undecidable. Basic papers on undecidable propositions, unsolvable problems and computable functions*.
- S. Feferman (1992), "Turing's 'oracle': From absolute to relative computability--and back", in *The Space of Mathematics* (J. Echeverria, et al., eds.)
- S. Feferman (2006), "Turing's thesis", in *Notices AMS* 53(#10)
- R. Herken (ed.) (1988), *The Universal Turing Machine. A half-century survey.*

Selected References (cont'd)

- A. Hodges (1983), *Alan Turing. The Enigma*
- S. C. Kleene (1952), *Introduction to Metamathematics*
- M. Lerman (1983), *Degrees of Unsolvability*
- H. Rogers (1967), *Theory of Recursive Functions and Effective Computability*
- G. E. Sacks (1990), *Higher Recursion Theory*
- R. I. Soare (2009), "Turing oracle machines, online computing, and three displacements in the theory of computation", *Annals of Pure and Applied Logic* 160.