Logic in Computer Science

Saeed Salehi

University of Tabriz

http://SaeedSalehi.ir/

IPM Summer School in Computer Science
7–15 July 2012, Tehran, Iran

Hilbert's Entscheidungsproblem = Decision Problem

Finding an ALGORITHM (or AL-KHWARIZMI):
Input: A (Mathematical) Statement.
Output: *YES* (if universally valid) *NO* (if not always valid).

How to write (code) mathematical statements (as input strings)?
Example from Al-Khwarizmi: If from a square, I subtract four of its roots and then take one-third of the remainder, finding this equal to four of the roots, the square will be 256.

Modern Notation: If I have $\frac{1}{3}(x^2 - 4x) = 4x$, then $x^2 = 256$.

More Modern: $\forall x[\frac{1}{3}(x^2 - 4x) = 4x \longrightarrow x^2 = 256]$.
This holds in the domain $\mathbb{N} - \{0\} = \{1, 2, 3, \cdots\}$ (but not in $\mathbb{N}$).
Indeed, $\mathbb{N} \models \forall x[\frac{1}{3}(x^2 - 4x) = 4x \longrightarrow x = 16 \vee x = 0]$.

Entscheidungsproblem = Decision Problem (Hilbert 1928)

Finding an ALGORITHM (or AL-KHWARIZMI):
Input: A (Mathematical) Statement.
Output: *YES* (if universally valid) *NO* (if not always valid).

How to write (code) mathematical statements?
Another Example from Al-Khwarizmi: What is the square which combined with ten of its roots will give a sum total of 39?
Modern Notation: What is (are) the solution(s) of $x^2 + 10x = 39$?
More Modern: $\forall x[x^2 + 10x = 39 \longrightarrow x = 3]$.
This holds in $\mathbb{N}$ but not in $\mathbb{Z}$:
$\mathbb{N} \models \forall x[x^2+10x = 39 \rightarrow x = 3], \mathbb{Z} \not\models \forall x[x^2+10x = 39 \rightarrow x = 3]$.
Indeed, $\mathbb{Z} \models \forall x[x^2 + 10x = 39 \longrightarrow x = 3 \vee x = -13]$.

Propositional Logic (SYNTAX)

Atomization of Language:

A Text (Book) consists of

Some Parts	a part consists of
Some Chapters	a chapter consists of
Some Sections	a section consists of
Some Paragraphs	a paragraph consists of
Some Sentences	a sentence consists of
Some Words	a word consists of
Some Letters	

Letters are the Atoms of Atomized Languages.

Propositional Logic (SYNTAX)

Atoms of Propositional Logic are Propositions:

$P_1, P_2, P_3, \cdots$ Without Meaning or Truth or Falsity

Connectives:

$\wedge$ Conjunction

$\vee$ Disjunction

$\rightarrow$ Implication

$\neg$ Negation

Example (Propositional Formula)

$P_1 \wedge (P_2 \vee P_3)$ $P_1 \rightarrow (P_2 \rightarrow P_1)$

$\neg P_1 \rightarrow (P_1 \rightarrow P_2)$ $(\neg P_1) \vee (P_1 \rightarrow P_2)$

Propositional Logic (SYNTAX)

PROBLEM: Show that the set of propositional formulas is a decidable subset of $\{\mathtt{P}, (,), \wedge, \vee, \rightarrow, \neg\}^*$ (by interpreting $\mathtt{P}_i = \mathtt{P}^i$).

Definition (Propositional Formulas)

The Set $\mathcal{L}_{\mathbb{P}}$ of Propositional Formulas is the Smallest Set that
(1) Contains All the Atoms: $\mathtt{P}^1, \mathtt{P}^2, \mathtt{P}^3, \cdots \in \mathcal{L}_{\mathbb{P}}$
(2) Is Closed Under $\wedge$, $\vee$, $\rightarrow$ and $\neg$:
if $\alpha, \beta \in \mathcal{L}_{\mathbb{P}}$ then $(\alpha \wedge \beta), (\alpha \vee \beta), (\alpha \rightarrow \beta), (\neg\alpha) \in \mathcal{L}_{\mathbb{P}}$.

Propositional Logic (SEMANTICS)

A Propositional Model is a FUNCTION $\nu : \texttt{Atoms} \to \{0,1\}$.

Extension to Composite Formulas:
$\nu(\alpha \wedge \beta) = \min\{\nu(\alpha), \nu(\beta)\}$,
$\nu(\alpha \vee \beta) = \max\{\nu(\alpha), \nu(\beta)\}$,
$\nu(\alpha \to \beta) = \max\{1 - \nu(\alpha), \nu(\beta)\}$,
$\nu(\neg\alpha) = 1 - \nu(\alpha)$.

- Tautology: for all ν we have $\nu(\alpha) = 1$;
- Contradiction: for no ν can we have $\nu(\alpha) = 1$;
- Satisfiable: for some ν we can have $\nu(\alpha) = 1$.

Propositional Logic (SEMANTICS)

PROBLEM: is the set of propositional tautologies decidable?

Lemma

The sets of Propositional Tautologies, Contradictions, and Satisfiable Formulas are Decidable.

Proof.

By High-School Truth-Table Algorithms $\cdots$. $\square$

Decidability Follows From the Finiteness of Models.

Finite Model Property

Propositional Logic (Laws)

$$(\alpha\wedge\alpha)\rightarrow\alpha \qquad (\alpha\vee\alpha)\rightarrow\alpha$$
$$\alpha\rightarrow(\alpha\wedge\alpha) \qquad \alpha\rightarrow(\alpha\vee\alpha)$$
$$(\alpha\wedge\beta)\rightarrow(\beta\wedge\alpha) \qquad (\alpha\vee\beta)\rightarrow(\beta\vee\alpha)$$
$$(\alpha\wedge(\beta\wedge\gamma))\rightarrow((\alpha\wedge\beta)\wedge\gamma) \qquad (\alpha\vee(\beta\vee\gamma))\rightarrow((\alpha\vee\beta)\vee\gamma)$$
$$((\alpha\wedge\beta)\wedge\gamma)\rightarrow(\alpha\wedge(\beta\wedge\gamma)) \qquad ((\alpha\vee\beta)\vee\gamma)\rightarrow(\alpha\vee(\beta\vee\gamma))$$
$$(\alpha\wedge(\beta\vee\gamma))\rightarrow((\alpha\wedge\beta)\vee(\alpha\wedge\gamma))$$
$$((\alpha\wedge\beta)\vee(\alpha\wedge\gamma))\rightarrow(\alpha\wedge(\beta\vee\gamma))$$
$$(\alpha\vee(\beta\wedge\gamma))\rightarrow((\alpha\vee\beta)\wedge(\alpha\vee\gamma))$$
$$((\alpha\vee\beta)\wedge(\alpha\vee\gamma))\rightarrow(\alpha\vee(\beta\wedge\gamma))$$
$$(\alpha\wedge(\alpha\vee\beta))\rightarrow\alpha \qquad (\alpha\vee(\alpha\wedge\beta))\rightarrow\alpha$$
$$\alpha\rightarrow(\alpha\wedge(\alpha\vee\beta)) \qquad \alpha\rightarrow(\alpha\vee(\alpha\wedge\beta))$$
$$\neg(\neg\alpha)\rightarrow\alpha \qquad \alpha\rightarrow\neg(\neg\alpha)$$
$$\neg(\alpha\wedge\beta)\rightarrow(\neg\alpha\vee\neg\beta) \qquad \neg(\alpha\vee\beta)\rightarrow(\neg\alpha\wedge\neg\beta)$$
$$(\neg\alpha\vee\neg\beta)\rightarrow\neg(\alpha\wedge\beta) \qquad (\neg\alpha\wedge\neg\beta)\rightarrow\neg(\alpha\vee\beta)$$

Propositional Logic (LAWS)

$$\alpha \to \alpha$$

$(\alpha \wedge \beta) \to \alpha$ $\qquad$ $\alpha \to (\alpha \vee \beta)$

$(\alpha \wedge \beta) \to \beta$ $\qquad$ $\beta \to (\alpha \vee \beta)$

$(\alpha \to \beta) \to (\neg\alpha \vee \beta)$ $\qquad$ $(\neg\alpha \vee \beta) \to (\alpha \to \beta)$

$(*)\ \alpha \to (\beta \to \alpha)$ $\qquad$ $(\neg\beta) \to (\beta \to \alpha)$

$$(*)\ [\alpha \to (\beta \to \gamma)] \to [(\alpha \to \beta) \to (\alpha \to \gamma)]$$

$(*)\ (\neg\beta \to \neg\alpha) \to (\alpha \to \beta)$ $\qquad$ $(\alpha \to \beta) \to (\neg\beta \to \neg\alpha)$

Propositional Logic (RULES)

$$(*)\ \frac{\alpha, \quad \alpha \to \beta}{\beta} \qquad\qquad \frac{\alpha \to \beta, \quad \beta \to \gamma}{\alpha \to \gamma}$$

Aristotle's Syllogism

This deduction cannot be formalized in Propositional Logic:

$$\frac{\text{All humans are mortal.} \qquad \text{Socrates is a human.}}{\therefore \qquad \text{Socrates is mortal.}}$$

We need some predicates for expressing `properties` of subjects, and some quantifiers to `quantify` (the number) of `the subjects satisfying some properties`.

Aristotle's Syllogistic: $\mathfrak{a}$ $\mathfrak{e}$ $\mathfrak{i}$:

$X\mathfrak{a}F$: Every X is F. $X\mathfrak{e}F$: No X is F. $X\mathfrak{i}F$: Some X is F.

Later was added $\mathfrak{o}$ – $X\mathfrak{o}F$: Some X is not F.

Aristotle's Syllogistic Rules

$$\text{First Figure}\ \frac{Y \,\Box\, Z, \quad X \bigcirc Y}{X \triangle Z} \qquad \text{example} : \frac{Y \mathfrak{a} Z,\ X \mathfrak{a} Y}{X \mathfrak{a} Z}$$

$$\text{Second Figure}\ \frac{Z \,\Box\, Y, \quad X \bigcirc Y}{X \triangle Z} \qquad \text{example} : \frac{Z \mathfrak{e} Y,\ X \mathfrak{i} Y}{X \mathfrak{i} Z}$$

$$\text{Third Figure}\ \frac{Y \,\Box\, Z, \quad Y \bigcirc X}{X \triangle Z} \qquad \text{example} : \frac{Y \mathfrak{e} Z,\ Y \mathfrak{a} X}{X \mathfrak{o} Z}$$

$$\text{Fourth Figure}\ \frac{Z \,\Box\, Y, \quad Y \bigcirc X}{X \triangle Z} \qquad \text{example} : \frac{Z \mathfrak{e} Y,\ Y \mathfrak{i} X}{X \mathfrak{o} Z}$$

Aristotle's Syllogistic Valid Rules

	□ ○ △
First Figure:	aaa, eae, aii, eio
Second Figure:	eae, aee, eio, aoo
Third Figure:	aai, iai, aii, eao, oao, eio
Fourth Figure:	aai, aee, iai, eao, eii

Simplified:

First Figure:	aaa, eae, aii, eio
Second Figure:	aoo
Third Figure:	aai, eao, oao

▷ Two of the above rules of Aristotle, are not valid – require some non-emptiness conditions.

▷ It is DECIDABLE to verify if a deduction is valid or not in Aristotle's Syllogistic.

(Unary) Predicate Logic (SYNTAX)

Variables (to be interpreted in some domains): $x, y, z, \cdots$
Predicates: $U(x), V(y), W(z), \cdots$
Quantifiers: $\forall, \exists$.

Definition ((Unary) Predicate Formulas)

(1) All expressions of the form $U(x)$ where U is a (unary) predicate and x is a variable, are formulas.
(2) Formulas are closed under $\wedge$, $\vee$, $\rightarrow$, $\neg$, $\forall x$ and $\exists y$.

Examples:

$$\forall x(H(x) \rightarrow M(x)) \wedge \forall x(S(x) \rightarrow H(x)) \longrightarrow \forall x(S(x) \rightarrow M(x))$$
$$\neg\exists y(C(y) \wedge B(y)) \rightarrow [\exists x(A(x) \wedge B(x)) \rightarrow \exists z(A(z) \wedge C(z))]$$
$$\forall x(U(x) \rightarrow \neg V(x)) \wedge \forall x(U(x) \rightarrow W(x)) \longrightarrow \exists y(W(y) \wedge \neg V(y))$$
$$\neg\exists x(C(x) \wedge B(x)) \rightarrow [\exists x(B(x) \wedge A(x)) \rightarrow \exists x(A(x) \wedge \neg C(x))]$$

(Unary) Predicate Logic (SEMANTICS)

Domain: A non-empty Arbitrary Set M.
Interpretation: For each predicate $U(x)$ a subset $U^{\mathcal{M}} \subseteq M$.
For a model $\mathcal{M} = \langle M, U, V, \cdots \rangle$ and for a formula φ the satisfaction of φ in $\mathcal{M}$ can be determined once the free variables of φ has been interpreted in M.

For example, for domain $= \mathbb{N}$, and predicates E (even numbers), P (prime numbers), Z (zero number), N (negative numbers) and S (square numbers) we have $\mathbb{N} \models \forall x(S(x) \rightarrow \neg P(x))$; $\mathbb{N} \models \neg\exists x N(x)$; $\mathbb{N} \models \exists x(P(x) \wedge E(x))$; $\mathbb{N} \models \exists x(Z(x) \wedge S(x))$.

Finite Model Property: If a unary predicate formula is satisfiable (has a model), then it is satisfiable in (has) a finite model.
So, Validity and Satisfiability are DECIDABLE in Aristotle's Syllogistic and Unary Predicate Logic.

Equational Logic (SYNTAX)

Logical Equality is an Equivalence and Congruence Relation:
$x = x$; $x = y \longrightarrow y = x$; $x = y \wedge y = z \longrightarrow x = z$;
$x = y \longrightarrow [\varphi(x) \to \varphi(y)]$ (Leibnitz's Principle).

Fix a set of Constant Symbols and a set of Function Symbols.

Definition (Terms)

(1) Every variable or constant is a term.
(2) Terms are closed under all the function symbols.

Example: for constants $0, 1$ function symbols $+, \cdot$ and variables x, y, z the following are terms (in the language $\{0, 1, +, \cdot\}$):
$x + (y \cdot 0)$, $x \cdot (y + 1)$, $(x \cdot 0) + (y \cdot 1)$, $(x + y) \cdot (x + z)$.

Equational Logic (SYNTAX)

Variables, Constants $\subseteq$ Terms.
f function symbol, $t_1, \cdots, t_n \in$Terms $\Rightarrow f(t_1, \cdots, t_n) \in$Terms.

Formulas of Equational Logic:
Like Propositional Logic, where Atomic Formulas are Equality of Two Terms. $t = u$ for terms t and u.

Example: $(x + y = x + z) \rightarrow (y = z)$; $x \cdot y = 1 \rightarrow x \neq 0$.

Exercise: For constant symbol 0, unary function symbol $-$ and binary function symbol $+$ prove that

$$[x + (y + z) = (x + y) + z] \wedge [x + 0 = x] \wedge [x + (-x) = 0] \longrightarrow [(y + x = x) \rightarrow y = 0].$$

Much of Algebra can be done in Equational Logic.

Equational Logic (SEMANTICS)

Domain: A non-empty Arbitrary Set M.
Interpretation: For each constant symbol c, a fixed $c^{\mathcal{M}} \in M$, and for each ($n-$ary) function symbol f, a fixed $f^{\mathcal{M}} : M^n \to M$.

Any (interpretation) function $h : X \to M$ from variables to domain, can be extended to $h : \text{Terms} \to M$

for constant c, $h(c) = c^{\mathcal{M}}$;
for function f, $h(f(t_1, \cdots, t_n)) = f^{\mathcal{M}}(h(t_1), \cdots, h(t_n))$.

For a model $\mathcal{M}$ with a function $h : X \to M$ and equation $t = u$, we write $\mathcal{M} \models_h t = u$ where $h(t) = h(u)$.

For a formula φ the notion $\mathcal{M} \models \varphi$ can be defined as satisfaction of the universal closure of φ in $\mathcal{M}$.

First–Order Logic (SYNTAX)

Fix a set of primitive constant, function, or relation symbols.
We will use constants $0, 1$; the functions $+, \cdot$; the relations $<, \leqslant$.
Terms are constructed from variables and constants by successive application of function symbols.
Examples: $0 + x,\ 1 \cdot (x + y),\ (x \cdot x) + y,$ algebraic expressions
Atomic Formulas are relations (including $=$) between terms:

$t = u$ or $t < u$ or $t \leqslant u$.

Formulas:

- Atomic Formulas
- $\neg\varphi$ (not φ)
- $\varphi \wedge \psi,\ \varphi \vee \psi,\ \varphi \to \psi$
- $\exists x\varphi(x),\ \forall x\varphi(x)$

Examples:

$\forall x\exists y[x = 2y \vee x = 2y + 1],\ \exists x\forall y[x + y = y],$
$\forall x[x + u = x],\ \forall y[y \cdot u = u],\ \forall z[z \cdot u = z],\ \exists z[z + x = y], \cdots$

First–Order Logic (SEMANTICS)

Fix a domain: a set to whose members the variables refer.
We will use the sets of numbers:

Natural ($\mathbb{N}$), Integer ($\mathbb{Z}$), Rational ($\mathbb{Q}$), Real ($\mathbb{R}$), Complex ($\mathbb{C}$).

Tarski's Definition of Truth defines satisfiability of a formula in a structure (by induction).

Examples:

- ▷ $\mathbb{N} \not\models \forall x \exists y (x + y = 0)$ but $\mathbb{Z} \models \forall x \exists y (x + y = 0)$.
- ▷ $\mathbb{Z} \not\models \forall x (x \neq 0 \rightarrow \exists y [x \cdot y = 1])$ but $\mathbb{Q} \models \forall x (x \neq 0 \rightarrow \exists y [x \cdot y = 1])$.
- ▷ $\mathbb{Q} \not\models \forall x (0 \leqslant x \rightarrow \exists y [y \cdot y = x])$ but $\mathbb{R} \models \forall x (0 \leqslant x \rightarrow \exists y [y \cdot y = x])$.
- ▷ $\mathbb{R} \not\models \exists x (x \cdot x + 1 = 0)$ but $\mathbb{C} \models \exists x (x \cdot x + 1 = 0)$.

Hilbert's Entscheidungsproblem = Decision Problem (AGAIN)

Finding an ALGORITHM (or AL-KHWARIZMI):
Input: A First–Order Sentence.
Output: *YES* (if universally valid) *NO* (if not always valid).

The history goes back to even G. Leibniz in the 17th century.

Theorem (Gödel's Completeness Theorem 1929)

The set of (universally) valid first–order sentences is computably enumerable (i.e., listable by an algorithm).

Computably Enumerable set A: an (input-free) algorithm $\mathcal{P}$ lists all members of A; i.e., $A = \texttt{output}(\mathcal{P})$.

Computably Decidable set A: an algorithm $\mathcal{P}$ decides on any input x whether $x \in A$ (outputs $\texttt{YES}$) or $x \notin A$ (outputs $\texttt{NO}$).

Gödel's Completeness Theorem

From An Axiomatization of (Logically) Valid First–Order Formulas:

- $\alpha \to (\beta \to \alpha)$ • $(\neg\beta \to \neg\alpha) \to (\alpha \to \beta)$
- $[\alpha \to (\beta \to \gamma)] \to [(\alpha \to \beta) \to (\alpha \to \gamma)]$
- $\forall x \varphi(x) \to \varphi(t)$ • $\varphi \to \forall x \varphi$ [x is not free in φ]
- $\forall x(\varphi \to \psi) \to (\forall x \varphi \to \forall x \psi)$

With the Modus Ponens Rule: $\dfrac{\varphi, \quad \varphi \to \psi}{\psi}$

All the Universally Valid Formulas CAN BE GENERATED.

Note: $A \vee B = (\neg A) \to B$, $A \wedge B = \neg[A \to (\neg B)]$, $\exists x \varphi = \neg \forall x \neg \varphi$.

Universally Valid Formulas: Computably Enumerable

All First-Order Formulas Can be Shown to be
(1) Universally Valid (in every structure) if and only if
(2) Deducible in FO Logic from the Axioms by the MP Rule.

Example:

- $\forall \mathbf{x}\big([(\alpha \to \beta) \to \alpha] \to \alpha\big)$
- $\exists y \forall x \big(\varphi(y) \to \varphi(x)\big)$

Mechanization of Reasoning:

Logical Deduction is an Automatic Generation.

(Semi–Decidable) $\not\equiv$ (Decidable)

Post–Kleene's Theorem: A Set is Computably Decidable if and only if Both it and its Complement are Computably Enumerable.

Algorithm ?

For defining COMPUTABILITY, we consider functions : $\mathbb{N}^k \to \mathbb{N}$.
Every computable object can be “coded” by a natural number:
Each ASCII code can be written by a string of 0’s and 1’s:

λ,	0,	1,	00,	01,	10,	11,	000,	001,	010,	011,	100,	101,	110,	$\cdots$
0,	1,	2,	3,	4,	5,	6,	7,	8,	9,	10,	11,	12,	13,	$\cdots$

$$(i_j \in \{0,1\}):\ \langle i_1, i_2, \cdots, i_k\rangle \mapsto (1i_1i_2\dots i_k)_2 - 1$$
$$(m \in \mathbb{N}):\ m \mapsto 1^{-1}\mathtt{bin}(n+1)$$

Finite Sequences of Natural Numbers can be coded in $\mathbb{N}$:

$$\langle n_0, n_1, \cdots, n_k\rangle \mapsto \prod_{i=0}^{k} \wp_i^{(n_i+1)}$$

where $\wp_i$ is the i^{th} prime number: $\wp_0 = 2$, $\wp_1 = 3$, $\wp_2 = 5$, $\cdots$.

Computable Functions : $\mathbb{N}^k \to \mathbb{N}$?

The Number of Functions : $\mathbb{N}^k \to \mathbb{N}$ is not countable:
there is no one-one correspondence between $\mathbb{N}$ and $\mathbb{N}^{\mathbb{N}}$:
for any mapping $\mathfrak{F} : \mathbb{N} \to \mathbb{N}^{\mathbb{N}}$ put $D_{\mathfrak{F}} : \mathbb{N} \to \mathbb{N}$, by
$D_{\mathfrak{F}}(x) = \mathfrak{F}(x)[x] + 1$. Then $D_{\mathfrak{F}} \notin \mathrm{Range}(\mathfrak{F})$ because
$D_{\mathfrak{F}}(m) \neq \mathfrak{F}(m)[m]$ and so $D_{\mathfrak{F}} \neq \mathfrak{F}(m)$ for any $m \in \mathbb{N}$.

But the Number of Computable Functions Must be Countable:
every program is a sequence of ASCII codes,
and thus is a string of 0's and 1's, which is a countable set.

`So, there are non-computable functions:` $\mathbb{N}^k \to \mathbb{N}$.

Goal: Characterizing the Computable Functions: $\mathbb{N}^k \to \mathbb{N}$.

Computable Functions : $\mathbb{N}^k \to \mathbb{N}$ = RECURSIVE FUNCTIONS

Recursive Functions Contain:

- Zero Constant Function $Z(x) = 0$
- Successor Function $S(x) = x + 1$
- Projection Functions $\pi_i^k(n_1, \cdots, n_k) = n_i$
- Addition Function $A(x, y) = x + y$
- Multiplication Function $M(x, y) = x \cdot y$
- Exponentiation Function $E(x, y) = x^y$
- Order Recognition Function $\chi_{\leqslant}(x, y) = 1 \vee 0 \; (x \leqslant y \vee x > y)$
- Prime Numbering Function $\wp(x) = x^{\text{th}}\texttt{prime number}$

and is Closed Under:

- The Composition of Functions
- Minimization of Functions

$$\big[\mu z. f(\mathbf{x}, z) = g(\mathbf{x}, z)\big](\mathbf{x}) = \min\{z \in \mathbb{N} \,|\, f(\mathbf{x}, z) = g(\mathbf{x}, z)\}$$

Computable Functions = Recursive Functions

By Experience: every (intuitionally) computable function has turned out to be recursive.

For example, it can be shown that recursive functions are closed under primitive recursion:

$$\mathrm{PR}^{f,g}(\mathbf{x},z): \quad \begin{array}{l} \mathrm{PR}^{f,g}(\mathbf{x},0) = f(\mathbf{x}) \\ \mathrm{PR}^{f,g}(\mathbf{x},z+1) = g(\mathbf{x},z,\mathrm{PR}^{f,g}(\mathbf{x},z)) \end{array}$$

Church's Thesis

Every (Intuitionally) Computable Function is Recursive.

Note that every recursive function is clearly computable.

Non-Computability

Some Recursive Functions may Never Halt (have outputs on some inputs); e.g., $D(x,y)=[\mu z.z+x=y]$ which is

$y-x$ if $x\leqslant y$ and $\texttt{undefined}$ if $x>y$.

$f(\mathbf{x})\downarrow$ means f of $\mathbf{x}$ is defined;

$f(\mathbf{x})\uparrow$ means f is $\texttt{undefined}$ on $\mathbf{x}$.

Recursive Functions can be encoded by natural numbers: Any description (proof) of a recursive function is a well-built sequence of $\langle Z,S,\pi_i^k,A,M,E,\chi_{\leqslant},\wp,\circ,\mu\rangle$ ($\circ$ stands for composition), and thus can be coded in $\mathbb{N}$.

Denote the (Gödel) code of the recursive function f by $\ulcorner f\urcorner$.

A Non-Computable Function

In Fact, the Entscheidungsproblem (Decision Problem of First-Order Logic) was the first one to be shown ALGORITHMICALLY UNSOLVABLE or Undecidable.

Theorem (Turing's Halting Problem)

There is no recursive function $\mathfrak{h}$ *such that for any* $f \in \mathrm{Recursive}$, $\mathfrak{h}(\ulcorner f \urcorner) = 1 \iff f(\ulcorner f \urcorner)\downarrow$ *and* $\mathfrak{h}(\ulcorner f \urcorner) = 0 \iff f(\ulcorner f \urcorner)\uparrow$.

Proof.

In that case $\mathfrak{g}(x) = \mu z.(z + \mathfrak{h}(x) = z)$ would be recursive too, for which we have $\mathfrak{g}(\ulcorner f \urcorner)\downarrow \iff f(\ulcorner f \urcorner)\uparrow$. Putting $\mathfrak{g}$ in place of f we get the contradiction $\mathfrak{g}(\ulcorner \mathfrak{g} \urcorner)\downarrow \iff \mathfrak{g}(\ulcorner \mathfrak{g} \urcorner)\uparrow$! $\square$

Some Non-Computable Functions

Theorem (Undecidability of Halting at Zero)

There is no recursive function $\mathfrak{h}$ *such that for any* $f \in \mathrm{Recursive}$*,*
$\mathfrak{h}(\ulcorner f \urcorner) = 1 \iff f(\mathbf{0})\downarrow$ *and* $\mathfrak{h}(\ulcorner f \urcorner) = 0 \iff f(\mathbf{0})\uparrow$*.*

Proof: By Reduction to Halting Problem.

In that case, $\mathfrak{g}$ defined by $\mathfrak{g}(\ulcorner f \urcorner) = \mathfrak{h}(\ulcorner f' \urcorner)$, where
$f'(x) = f(\ulcorner f \urcorner)$, would be recursive with the property
$\mathfrak{g}(\ulcorner f \urcorner) = 1 \iff f(\ulcorner f \urcorner)\downarrow$ and $\mathfrak{g}(\ulcorner f \urcorner) = 0 \iff f(\ulcorner f \urcorner)\uparrow$.
Contradiction by Turing's Theorem on the Undecidability of the Halting Problem. $\square$

Alan M. Turing, "On Computable Numbers, With an Application to the Entscheidungsproblem",
Proceedings of the London Mathematical Society, Series 2, 42 (1937), pp 230–265.
`http://dx.doi.org/10.1112/plms/s2-42.1.230`

Representability of Recursive Functions

Let $\mathcal{L} = \{0, S, \wp, +, \cdot, E, \leqslant, =\}$ be a First-Order Language

(0 constant, $S, \wp$ unary functions, $+, \cdot, E$ binary functions, $\leqslant, =$ binary relations).

The Logic Allows Recursive Functions to be $\mathcal{L}-$Representable.

Theorem (Representability of a Function)

Any recursive function $f(\mathbf{x})$ is representable in $\mathcal{L}$; i.e., there exists an $\mathcal{L}-$formula $\Psi_f(\mathbf{x}, y)$ such that for any $\mathbf{m}, n \in \mathbb{N}$, $f(\mathbf{m}) = n$ if and only if $\Psi_f(S^{\mathbf{m}}0, S^n 0)$ is logically valid.

$\Psi_f = \bigwedge\!\!\!\bigwedge Q \rightarrow \psi_f$: $\psi_Z(x, y) \equiv (y = 0)$; $\psi_+(x, y, z) \equiv (z = x + y)$;
$\psi_{\chi_\leqslant}(x, y, z) \equiv (x \leqslant y \wedge z = 1) \vee (\neg x \leqslant y \wedge z = 0)$;
$\psi_{f \circ \mathbf{g}}(\mathbf{x}, y) \equiv (\exists \mathbf{z}[\psi_{\mathbf{g}}(\mathbf{x}, \mathbf{z}) \wedge \psi_f(\mathbf{z}, y)])$;
$\psi_{\mu z.f=g}(\mathbf{x}, y) \equiv \big(\exists u[\psi_f(\mathbf{x}, y, u) \wedge \psi_g(\mathbf{x}, y, u)] \wedge$
$\forall v[v < y \rightarrow \exists w, w'(\psi_f(\mathbf{x}, v, w) \wedge \psi_g(\mathbf{x}, v, w') \wedge \neg w = w')]\big)$.

Undecidability of Logical Validity (Entscheidungsproblem)

Since the problem of halting at zero for recursive functions is not decidable, then neither is logical validity for $\mathcal{L}-$formulas.

Theorem (Church's Theorem – Proved Also by Turing (1936))

There Is No Algorithm For Deciding Whether A Given ($\mathcal{L}-$)Formula Is Logically Valid.

Proof.

For $f \in \mathrm{Recursive}$: $f(\mathbf{0})\downarrow \iff [\exists y \Psi_f(\mathbf{0}, y)$ is logically valid]. $\square$

Corollary (Gödel's Incompleteness Theorem 1931)

The Theory $(\mathbb{N}, \mathcal{L})$ Is Un–Decidable.

Validity: Un–Decidable & Semi–Decidable

The Arithmetical Theory Q Contains Axioms Like:

- $\forall x[S(x) \neq 0]$ • $\forall x \forall y[S(x) = S(y) \rightarrow x = y]$

Which Can Have Infinite Models Only (do not allow finite models). None–(Finite Model Property) Does Not Imply Undecidability (Finite Model Property Implies Decidability). Undecidability of Arithmetic Is Deeper.

Thus, though the set of Universally Valid formulas is Computably Enumerable, there is no algorithm for deciding whether a given first–order formula is valid in all structures or not.

So, Hilbert's Entscheidungsproblem is (algorithmically) Un–Solvable; though he did expect an algorithm.

► A Good Outcome: Introducing Turing Machines – the grand grandfather of today's modern computers.

✪ ❀ Thank You ! ❀ ✪

✳ ❁ ❁ ❁ ❁ ❁ ❁ ❁ ❁ ❁ ✳

For Listening ...

❁ ❁ ❁