CM0081 Automata and Formal Languages
Undecidable Problems

Andrés Sicard-Ramírez

Universidad EAFIT

Semester 2024-1

Introduction

Undecidable Problems

There are undecidable problems in different domains:

- Analysis
- Logic
- Matrices
- Topology
- Physics
- Among other

Normal Forms for the Lambda Calculus

Alonzo Church (1903 – 1995)[†]

[†]Figures sources: History of computers, Wikipedia and MacTutor History of Mathematics.

Normal Forms for the Lambda Calculus

Some remarks about the λ-calculus

- A formal system invented by Church around 1930s.
- The goal was to use the λ-calculus in the foundation of mathematics.
- Intended for studying functions and recursion.
- Computability model.
- A free-type functional programming language.
- λ-notation (e.g. anonymous functions and currying).

Normal Forms for the Lambda Calculus

Application

Application of the function M to argument N is denoted by MN (juxtaposition).

Normal Forms for the Lambda Calculus

Application

Application of the function M to argument N is denoted by MN (juxtaposition).

Abstraction

> *'If M is any formula containing the variable x, then $\lambda x[M]$ is a symbol for the function whose values are those given by the formula.' [Church 1932, p. 352]*

Normal Forms for the Lambda Calculus

Currying

> *'Adopting a device due to Schönfinkel, we treat a function of two variables as a function of one variable whose values are functions of one variable, and a function of three or more variables similarly.' [Church 1932, p. 352]*

Such device is called **currying** after Haskell Curry.

(continued on next slide)

Normal Forms for the Lambda Calculus

Currying (continuation)

Let $g : X \times Y \to Z$ be a function of two variables. We can define two functions f_x and f:

$$f_x : Y \to Z \qquad\qquad f : X \to (Y \to Z)$$
$$f_x = \lambda y.g(x, y), \qquad\qquad f = \lambda x.f_x.$$

Then $(f\, x)\, y = f_x\, y = g(x, y)$. That is, the function of two variables

$$g : X \times Y \to Z$$

is represented as the higher-order function

$$f : X \to (Y \to Z).$$

Normal Forms for the Lambda Calculus

Definition

Let V be a denumerable set of variables. The set of **$\boldsymbol{\lambda}$-terms**, denoted by Λ, is inductively defined by

$$x \in V \Rightarrow x \in \Lambda \qquad \text{(variable)}$$

$$M, N \in \Lambda \Rightarrow (MN) \in \Lambda \qquad \text{(application)}$$

$$M \in \Lambda, x \in V \Rightarrow (\lambda x.M) \in \Lambda \qquad \text{(λ-abstraction)}$$

Normal Forms for the Lambda Calculus

Definition

Let V be a denumerable set of variables. The set of $\boldsymbol{\lambda}$**-terms**, denoted by Λ, is inductively defined by

$$
\begin{aligned}
x \in V &\Rightarrow x \in \Lambda && \text{(variable)} \\
M, N \in \Lambda &\Rightarrow (MN) \in \Lambda && \text{(application)} \\
M \in \Lambda, x \in V &\Rightarrow (\lambda x.M) \in \Lambda && \text{(λ-abstraction)}
\end{aligned}
$$

Observation

Usually, the set of λ-terms is defined by an abstract grammar like

$$t ::= x \mid t\,t \mid \lambda x.t$$

Normal Forms for the Lambda Calculus

Conventions

- λ-term variables will be denoted by $x, y, z, \ldots$.
- λ-terms will be denoted by $M, N, \ldots$.

Normal Forms for the Lambda Calculus

Conventions

- λ-term variables will be denoted by $x, y, z, \dots$.
- λ-terms will be denoted by $M, N, \dots$.

Example

Whiteboard.

Normal Forms for the Lambda Calculus

Conventions and syntactic sugar

- Outermost parentheses are not written.
- Application has higher precedence, i.e.,

$$\lambda x.MN := (\lambda x.(MN)).$$

- Application associates to the left, i.e.,

$$MN_1N_2\dots N_n := (\dots((MN_1)N_2)\dots N_n).$$

- Abstraction associates to the right, i.e.,

$$\begin{aligned}\lambda x_1x_2\dots x_n.M &:= \lambda x_1.\lambda x_2.\dots\lambda x_n.M\\ &:= (\lambda x_1.(\lambda x_2.(\dots(\lambda x_n.M)\dots))).\end{aligned}$$

Normal Forms for the Lambda Calculus

Definition

A variable x occurs **free** in M if x is not in the scope of λx. Otherwise, x occurs **bound**.

Definition

The **set of free variables in** $\mathbf{M}$, denoted by $\mathrm{FV}(M)$, is inductively defined by

$$\begin{aligned} \mathrm{FV}(x) &:= \{x\}, \\ \mathrm{FV}(MN) &:= \mathrm{FV}(M) \cup \mathrm{FV}(N), \\ \mathrm{FV}(\lambda x.M) &:= \mathrm{FV}(M) - \{x\}. \end{aligned}$$

Normal Forms for the Lambda Calculus

Definition

A variable x occurs **free** in M if x is not in the scope of λx. Otherwise, x occurs **bound**.

Definition

The **set of free variables in $\mathbf{M}$**, denoted by $\mathrm{FV}(M)$, is inductively defined by

$$\begin{aligned} \mathrm{FV}(x) &:= \{x\}, \\ \mathrm{FV}(MN) &:= \mathrm{FV}(M) \cup \mathrm{FV}(N), \\ \mathrm{FV}(\lambda x.M) &:= \mathrm{FV}(M) - \{x\}. \end{aligned}$$

Notation

The symbol '$\equiv$' denotes the syntactic identity.

Normal Forms for the Lambda Calculus

Definition

The result of **substituting $\mathbf{N}$ for every free occurrence of $\mathbf{x}$ in $\mathbf{M}$**, and changing bound variables to avoid clashes, denoted by $M[\,x/N\,]$, is defined by [Hindley and Seldin 2008, Definition 1.12]

$$
\begin{aligned}
x[\,x/N\,] &:= N,\\
y[\,x/N\,] &:= y,\ \text{if } y \not\equiv x,\\
(PQ)[\,x/N\,] &:= (P[\,x/N\,]\,Q[\,x/N\,]),\\
(\lambda x.P)[\,x/N\,] &:= \lambda x.P,\\
(\lambda y.P)[\,x/N\,] &:= \lambda y.P,\ \text{if } y \not\equiv x \text{ and } x \notin \mathrm{FV}(P),\\
(\lambda y.P)[\,x/N\,] &:= \lambda y.P[\,x/N\,],\ \text{if } y \not\equiv x, x \in \mathrm{FV}(P) \text{ and } y \notin \mathrm{FV}(N),\\
(\lambda y.P)[\,x/N\,] &:= \lambda z.P[\,x/N\,][\,y/z\,],\ \text{if } y \not\equiv x, x \in \mathrm{FV}(P) \text{ and}\\
&\qquad y \in \mathrm{FV}(N),
\end{aligned}
$$

where in the last equation, the variable z is chosen such that $z \notin \mathrm{FV}(NP)$.

Normal Forms for the Lambda Calculus

Definition

The functional behaviour of the λ-calculus is formalised through of their reduction/conversion rules. The β-**reduction rule** is defined by

$$(\lambda x.M)N \to_\beta M[\,x/N\,].$$

Normal Forms for the Lambda Calculus

Definition

The functional behaviour of the λ-calculus is formalised through of their reduction/conversion rules. The β-**reduction rule** is defined by

$$(\lambda x.M)N \to_\beta M[\,x/N\,].$$

Examples

- $(\lambda y.yy)x \to_\beta xx$

Normal Forms for the Lambda Calculus

Definition

The functional behaviour of the λ-calculus is formalised through of their reduction/conversion rules. The β-**reduction rule** is defined by

$$(\lambda x.M)N \rightarrow_{\beta} M[\,x/N\,].$$

Examples

- $(\lambda y.yy)x \rightarrow_{\beta} xx$
- $(\lambda x.(\lambda y.yx)z)v \rightarrow_{\beta} (\lambda y.yv)z \rightarrow_{\beta} zv$

Normal Forms for the Lambda Calculus

Definition

The functional behaviour of the λ-calculus is formalised through of their reduction/conversion rules. The β-**reduction rule** is defined by

$$(\lambda x.M)N \to_\beta M[\,x/N\,].$$

Examples

- $(\lambda y.yy)x \to_\beta xx$
- $(\lambda x.(\lambda y.yx)z)v \to_\beta (\lambda y.yv)z \to_\beta zv$
- Let Ω be $(\lambda x.xx)(\lambda x.xx)$, then $\Omega \to_\beta \Omega \to_\beta \cdots$

Normal Forms for the Lambda Calculus

Definition

A β-**redex** is a λ-term of the form $(\lambda x.M)N$.

Definition

A λ-term which contains no β-redex is in β-**normal form** (β-nf).

Definition

A λ-term N **is a** β-**nf of** M (or M **has the** β-**nf** M) iff N is a β-nf and $M =_\beta N$, where $=_\beta$ is the equivalence relation generated by the reflexive and transitive closure of $\to_\beta$.

Example

Whiteboard.

Normal Forms for the Lambda Calculus

Theorem

The set

$$\mathrm{NF} := \{\, M \in \Lambda \mid M \text{ has normal form} \,\}$$

is not recursive (i.e. undecidable) [Church 1935, 1936].

Observation

This was the first undecidable set ever.

Normal Forms for the Lambda Calculus

Observation

For proving that the set NF is undecidable we need an encoding and a version of Rice's theorem for λ-calculus.

Normal Forms for the Lambda Calculus

Observation

For proving that the set NF is undecidable we need an encoding and a version of Rice's theorem for λ-calculus.

Gödel numbering

The Gödel numbering for the λ-terms is defined by

$$\begin{aligned} \# : \Lambda &\to \mathbb{N} \\ \#(x_i) &= 2^i, \\ \#(\lambda x_i.M) &= 3^i 5^{\#(M)}, \\ \#(MN) &= 7^{\#(M)} 11^{\#(N)}. \end{aligned}$$

Normal Forms for the Lambda Calculus

Theorem (Rice's theorem for the λ-calculus)

Let $A \subset \Lambda$ such as A is non-trivial (i.e. $A \neq \emptyset$ and $A \neq \Lambda$). Suppose that A is closed under $=_\beta$ (i.e. $M \in A$ and $M =_\beta N$ then $N \in A$). Then the set A is undecidable, that is,

$$\{\, \#(M) \mid M \in A \,\} \quad \text{is undecidable.}$$

See [Barendregt 1990].

Normal Forms for the Lambda Calculus

Theorem (Rice's theorem for the λ-calculus)

Let $A \subset \Lambda$ such as A is non-trivial (i.e. $A \neq \emptyset$ and $A \neq \Lambda$). Suppose that A is closed under $=_\beta$ (i.e. $M \in A$ and $M =_\beta N$ then $N \in A$). Then the set A is undecidable, that is,

$$\{\, \#(M) \mid M \in A \,\} \quad \text{is undecidable.}$$

See [Barendregt 1990].

Proof (undecidability of NF)

Since the set NF is not trivial and it is closed under $=_\beta$, the set is undecidable. ■

The *Entscheidungsproblem*

The problem

The *Entscheidungsproblem* (decision problem) can be stated in three equivalent ways [Davis 2013, p. 49]:

(i) *Find an algorithm to determine whether a given sentence of first order logic is valid, that is, true regardless of what specific objects and relationships are being reasoned about.*

(ii) *Find an algorithm to determine whether a given sentence of first order logic is satisfiable, that is, true for some specific objects and relationships.*

(iii) *Find an algorithm to determine given some sentences of first order logic regarded as premises and another sentence, being a desired conclusion, whether that conclusion is provable from the premises using the rules of proof for first order logic.*

The *Entscheidungsproblem*

Historical remark

The *Entscheidungsproblem* was posed by Hilbert and Ackermann in 1928 [Hilbert and Ackermann 1950].

The *Entscheidungsproblem*

Historical remark

The *Entscheidungsproblem* was posed by Hilbert and Ackermann in 1928 [Hilbert and Ackermann 1950].

Negative answer

Church [1935, 1936] and Turing [1936–1937] gave a negative answer to the *Entscheidungsproblem* from the undecidability of the normal forms for the λ-calculus and the halting problem for Turing machines, respectively.

Post's Correspondence Problem (PCP)

An instance of the PCP

An instance of PCP consist of two lists of equal length

$$A = w_1, \ldots, w_k \quad \text{and} \quad B = x_1, \ldots, x_k$$

of strings over an alphabet Σ.

(continued on next slide)

Post's Correspondence Problem (PCP)

An instance of the PCP (continuation)

We say that the previous instance of PCP has a solution, if there is a sequence of one or more integers

$$i_1, \dots, i_m, \text{ with } m \geq 1$$

that, when interpreted as indexes for strings in the A and B lists, yield the same string, i.e.

$$w_{i_1} \cdots w_{i_m} = x_{i_1} \cdots x_{i_m}.$$

The sequence

$$i_1, \dots, i_m$$

is called a solution of the instance of PCP.

Post's Correspondence Problem (PCP)

The problem

Given an instance of PCP, tell whether this instance has a solution.

Post's Correspondence Problem (PCP)

The problem

Given an instance of PCP, tell whether this instance has a solution.

Example 9.13

An instance of the PCP:

	List A	List B
i	w_i	x_i
1	1	111
2	10111	10
3	10	0

Solution: $2, 1, 1, 3$, $m = 4$.

Post's Correspondence Problem (PCP)

Undecidability proof

The PCP problem is undecidable [Post 1946]. Hopcroft, Motwani and Ullman [2007] shows the undecidability via a reduction of $\mathrm{L_u}$ to PCP.

The Mortal Matrix Problem (MMP)

The problem

Let S be a finite set of $n \times n$ matrices with integer entries. To determine whether the zero matrix belongs to the semigroup generated by S, i.e. to determine whether the matrices in S can be multiplied in some order, possibly with repetitions, to yield the zero matrix.

The Mortal Matrix Problem (MMP)

The problem

Let S be a finite set of $n \times n$ matrices with integer entries. To determine whether the zero matrix belongs to the semigroup generated by S, i.e. to determine whether the matrices in S can be multiplied in some order, possibly with repetitions, to yield the zero matrix.

Some undecidable instances

The MMP is undecidable for a set of seven 3×3 matrices, or a set of two 21×21 matrices [Halava, Harju and Hirvensalo 2007].

The Mortal Matrix Problem (MMP)

The problem

Let S be a finite set of $n \times n$ matrices with integer entries. To determine whether the zero matrix belongs to the semigroup generated by S, i.e. to determine whether the matrices in S can be multiplied in some order, possibly with repetitions, to yield the zero matrix.

Some undecidable instances

The MMP is undecidable for a set of seven 3×3 matrices, or a set of two 21×21 matrices [Halava, Harju and Hirvensalo 2007].

Undecidability proof

Reduction of PCP to MMP.

Hilbert's Tenth Problem

Definition

A **Diophantine equation** is an equation of the form

$$D(x_1, \ldots, x_k) = 0,$$

where D is a polynomial with integer coefficients.

Hilbert's Tenth Problem

Definition

A **Diophantine equation** is an equation of the form

$$D(x_1, \dots, x_k) = 0,$$

where D is a polynomial with integer coefficients.

The problem (in present terminology)

> *'Given a Diophantine equation with any number of unknowns: To devise a process according to which it can be determined by a finite number of operations whether the equation has non-negative integer solutions.' [Sicard, Ospina and Vélez 2006, p. 12542]*

Hilbert's Tenth Problem

Definition

A **Diophantine equation** is an equation of the form

$$D(x_1, \ldots, x_k) = 0,$$

where D is a polynomial with integer coefficients.

The problem (in present terminology)

> *'Given a Diophantine equation with any number of unknowns: To devise a process according to which it can be determined by a finite number of operations whether the equation has non-negative integer solutions.' [Sicard, Ospina and Vélez 2006, p. 12542]*

Undecidability proof

A set is recursively enumerable if and only if it is Diophantine [Matiyasevich 1993].

Undecidable Problems in Physics

Some undecidable problems

‘Physics is also full of non-computable problems. The undecidability of the presence of chaos in classical Hamiltonian systems has been established[33]. The problem whether a boolean combination of subspaces (including negations) is reachable by a quantum automation was proved to be undecidable[34]. The question whether a quantum system is gapless also cannot be decided by an algorithm[35–37]. Whether a many-body model is frustration-free is undecidable as well[38]. Smith (Sec. 6 of[39]) identified a striking physical consequence of the Hilbert’s tenth problem that ground state energies and half-life times of excited states are, strictly speaking, non-computable for many-body systems. A variety of seemingly simple problems in quantum information theory has been shown not to be decidable[40]. The question whether a sequence of outcomes of some sequential measurement cannot be observed is undecidable in quantum mechanics, whereas it is decidable in classical physics[41]. In this case, the algorithmic undecidability turned out to be the signature of quantumness.’ [Bondar and Pechen 2020, p. 2]

References

Barendregt, H. (1990). Functional Programming and Lambda Calculus. In: Handbook of Theoretical Computer Science. Ed. by van Leeuwen, J. Vol. B. Formal Models and Semantics. MIT Press. Chap. 7. DOI: 10.1016/B978-0-444-88074-1.50012-3 (cit. on pp. 25, 26).

Bondar, D. I. and Pechen, A. N. (2020). Uncomputability and Complexity of Quantum Control. Scientific Reports 10, p. 1195. DOI: 10.1038/s41598-019-56804-1 (cit. on p. 41).

Church, A. (1932). A Set of Postulates for the Foundation of Logic. Annals of Mathematics 33.2, pp. 346–366. DOI: 10.2307/1968337 (cit. on pp. 5–7).

— (1935). An Unsolvable Problem of Elementary Number Theory. Preliminar Report (Abstract). Bulletin of the American Mathematical Society 41.5, pp. 332–333. DOI: 10.1090/S0002-9904-1935-06102-6 (cit. on pp. 22, 28, 29).

— (1936). An Unsolvable Problem of Elementary Number Theory. American Journal of Mathematics 58.2, pp. 345–363. DOI: 10.2307/2371045 (cit. on pp. 22, 28, 29).

References

Davis, M. (2013). Three Proofs of the Unsolvability of the Entscheidungsproblem. In: Alan Turing. His Work and Impact. Ed. by Cooper, S. B. and van Leeuwen, J. Elsevier, pp. 49–52 (cit. on p. 27).

Halava, V., Harju, T. and Hirvensalo, M. (2007). Undecidability Bounds for Integer Matrices Using Claus Instances. International Journal of Foundations of Computer Science 18.5, pp. 931–948. DOI: 10.1142/S0129054107005066 (cit. on pp. 35–37).

Hilbert, D. and Ackermann, W. [1938] (1950). Principles of Mathematical Logic. 2nd ed. Translation of the second edition of *Grundzüge der Theoretischen Logik*, Springer, 1938. Translated by Lewis M. Hammond, George G. Leckie and F. Steinhardt. Edited and with notes by Robert E. Luce. Chelsea Publising Company (cit. on pp. 28, 29).

Hindley, J. R. and Seldin, J. P. (2008). Lambda-Calculus and Combinators. An Introduction. Cambridge University Press (cit. on p. 16).

Hopcroft, J. E., Motwani, R. and Ullman, J. D. [1979] (2007). Introduction to Automata Theory, Languages, and Computation. 3rd ed. Pearson Education (cit. on p. 34).

Matiyasevich, Y. V. (1993). Hilbert's Tenth Problem. MIT Press (cit. on pp. 38–40).

References

Post, E. (1946). A Variant of a Recursively Unsolvable Problem. Bulletin of the American Mathematical Society 52, pp. 264–268. DOI: 10.1090/S0002-9904-1946-08555-9 (cit. on p. 34).

Sicard, A., Ospina, J. and Vélez, M. (2006). Quantum Hypercomputation Based on the Dynamical Algebra $\mathfrak{su}(1,1)$. J. Phys. A: Math. Gen. 39.40, pp. 12539–12558. DOI: 10.1088/0305-4470/39/40/018 (cit. on pp. 38–40).

Turing, A. M. (1936–1937). On Computable Numbers, with an Application to the Entscheidungsproblem. Proceeding of the London Mathematical Society s2-42, pp. 230–265. DOI: 10.1112/plms/s2-42.1.230 (cit. on pp. 28, 29).