Advanced Topics in Theoretical Computer Science

Part 4: Computability and (Un-)Decidability

20.12.2017

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Last time

- Recall: Turing machines and Turing computability
- Register machines (LOOP, WHILE, GOTO)
- Recursive functions
- The Church-Turing Thesis
- Computability and (Un-)decidability
- Complexity
- Other computation models: e.g. Büchi Automata

Today

- Recapitulation: Turing machines and Turing computability
- Recursive functions
- Register machines (LOOP, WHILE, GOTO)
- The Church-Turing Thesis
- **Computability and (Un-)decidability**
- Complexity
- Other computation models: e.g. Büchi Automata

Plan

- Recall: Acceptance and Decidability
- Recall: Undecidability results
  - The halting problem
  - Undecidability proofs via reduction
- The theorem of Rice (Variant 1)

Acceptance and Decidability

Acceptance

A DTM $\mathcal{M}$ accepts a language L if

- for every input word $w \in L$, $\mathcal{M}$ halts;
- for every input word $w \notin L$, $\mathcal{M}$ computes infinitely or hangs.

Deciding

A DTM $\mathcal{M}$ decides a language L if

- for every input word $w \in L$, $\mathcal{M}$ halts with band contents Y (yes)
- for every input word $w \notin L$, $\mathcal{M}$ halts with band contents N (no)

Acceptance and Decidability

Definition (Decidable language)

Let L be a language over Σ_0 with $\#, Y, N \notin \Sigma_0$; $\mathcal{M}=(K, \Sigma, \delta, s)$ a DTM with $\Sigma_0 \subseteq \Sigma$.

- $\mathcal{M}$ decides L if for all $w \in \Sigma_0^*$: $s, \#w\underline{\#} \vdash_{\mathcal{M}}^* \begin{cases} h, \#Y\underline{\#} & \text{if } w \in L \\ h, \#N\underline{\#} & \text{if } w \notin L \end{cases}$
- L is called decidable if there exists a DTM which decides L.

Definition (Acceptable language)

Let L be a language over Σ_0 with $\#, Y, N \notin \Sigma_0$; $\mathcal{M}=(K, \Sigma, \delta, s)$ a DTM with $\Sigma_0 \subseteq \Sigma$.

- $\mathcal{M}$ accepts a word $w \in \Sigma_0^*$ if $\mathcal{M}$ always halts on input w.
- $\mathcal{M}$ accepts the language L if for all $w \in \Sigma_0^*$, $\mathcal{M}$ accepts w iff $w \in L$.
- L is called acceptable (or semi-decidable) if there exists a DTM which accepts L.

Recursively enumerable

Definition (Recursively enumerable language)
Let L be a language over Σ_0 with $\#, Y, N \notin \Sigma_0$. Let $\mathcal{M} = (K, \Sigma, \delta, s)$ be a DTM with $\Sigma_0 \subseteq \Sigma$.

- $\mathcal{M}$ enumerates L if there exists a state $q_B \in K$ (the blink state) such that:

$$L = \{w \in \Sigma_0^* \mid \exists u \in \Sigma^*; s, \underline{\#} \vdash_{\mathcal{M}}^* q_B, \#w\underline{\#}u\}$$

- L is called recursively enumerable if there exists a DTM $\mathcal{M}$ which enumerates L.

Recursively enumerable

Attention: recursively enumerable $\neq$ enumerable!

- **L enumerable:** there exists a surjective map of the natural numbers onto L.
- **L recursively enumerable:** the surjective map can be computed by a TM.

Because of the finiteness of the words and of the alphabet, all languages are enumerable. But not all languages are recursively enumerable.

$\mapsto$ Set of all languages is not enumerable; Turing machines can be enumerated.

Attention: recursively enumerable $\neq$ decidable!

Examples: The following sets are recursively enumerable, but not decidable:

- The set of the Gödelisations of all halting Turing machines.
- The set of all terminating programs.
- The set of all valid formulae in predicate logic.

Acceptable/Recursively enumerable/Decidable

Theorem (Acceptable = Recursively enumerable)
A language is recursively enumerable iff it is acceptable.

Proposition
Every decidable language is acceptable.

Proposition
The complement of any decidable language is decidable.

Proposition (Characterisation of decidability)
A language L is decidable iff L and its complement are acceptable.

Recursively enumerable = Type 0

Formal languages are of type 0 if they can be generated by arbitrary grammars (no restrictions).

Proposition

The recursively enumerable languages (i.e. the languages acceptable by DTMs) are exactly the languages generated by arbitrary grammars (i.e. languages of type 0).

Undecidability results

Undecidability results: Proof via reduction

Given L_1, L_2 languages

L_1 known to be undecidable

To show L_2 undecidable

Idea

Assume L_2 decidable. Let M_2 be a TM which decides L_2.
We show that then we can construct a TM which decides L_1.

For this, we have to find a computable function f which transforms all elements of L_1 (and only the elements of L_1) into elements of L_2, i.e.

$$\forall w (w \in L_1 \text{ iff } f(w) \in L_2)$$

Let M_f be the TM which computes f.
Construct $M_1 = M_f M_2$. Then M_1 decides L_1.

TM; Gödelisation; Gödel numbers

Gödelisation: Method for assigning with every Turing machine M a number or a word (Gödel number or Gödel word) such that the Turing machine can be effectively reconstructed from that number (or word).

Gödel word of a TM M: $G(M)$

Gödel numbers. We sketched a possibility of associating with every Turing Machine M a unique Gödel number $\langle M \rangle \in \mathbb{N}$ such that the coding function and the decoding function are primitive recursive. Similarly for configurations of Turing machines.

Encoding words as natural numbers: If $\Sigma = \{a_0, a_1, \ldots, a_m\}$ and $w = a_{i_1} \ldots a_{i_n}$ is a word over Σ then $\langle w \rangle_l = \langle i_1, \ldots, i_n \rangle = \prod_{j=1}^{n} p(j)^{i_j}$

This shows, in particular, that we can represent w.l.o.g. words as natural numbers and languages as sets of natural numbers.

Undecidability of the halting problem

Theorem: $HALT = \{\langle G(\mathcal{M}), w\rangle \mid \mathcal{M} \text{ halts on input } w\}$ is not decidable.

Undecidability of the halting problem

Theorem: $HALT = \{\langle G(\mathcal{M}), w\rangle \mid \mathcal{M}$ halts on input $w\}$ is not decidable.

Theorem. $K = \{G(M) \mid M$ halts for input $G(M)\}$
is acceptable but undecidable.

Proof: Similar to the proof of the halting problem.

Undecidability proofs: Example

Theorem: $HALT = \{\langle G(\mathcal{M}), w\rangle \mid \mathcal{M}$ halts on input $w\}$ is not decidable.

Theorem. $K = \{G(M) \mid M$ halts for input $G(M)\}$
is acceptable but undecidable.

Reformulation using numbers instead of words:

Gödelization $\mapsto$ Gödel numbers

Let $M_0, M_1, \dots, M_n, \dots$ be an enumeration of all Turing Machines

M_n is the TM with Gödel number n.

$$\begin{array}{rcl} HALT & = & \{\langle n, i\rangle \mid M_n \text{ halts on input } i\} \\ K & = & \{n \mid M_n \text{ halts on input } n\} \end{array}$$

Undecidability proofs: Example

Theorem. $H_0 = \{n \mid M_n \text{ halts for input } 0\}$ is undecidable.

Proof: We show that K can be reduced to H_0, i.e. that there exists a TM computable function $f : \mathbb{N} \to \mathbb{N}$ such that $\quad i \in K \quad$ iff $\quad f(i) \in H_0$.

Want: $f(i) = j \quad$ iff $\quad$ (M_i halts for input i iff M_j halts for input 0).

For every i there exists a TM A_i s.t.: $s, \#\underline{\#} \vdash^*_{A_i} h, \#|^i\underline{\#}$.
Let M_K be the TM which accepts K.

We define $f(i) := j$ where $\quad j$ is the Gödel number of $M_j = A_i M_K$.
f is clearly TM computable. We show that f has the desired property:

$$
\begin{aligned}
f(i) = j \in H_0 \quad & \text{iff} \quad M_j = A_i M_K \text{ halts for input } 0 \ (\#\underline{\#}) \\
& \text{iff} \quad M_K \text{ halts for input } i \quad \text{iff} \quad i \in K.
\end{aligned}
$$

The theorem of Rice

Preliminaries:

Let M_n be the TM with Gödel number n.

If M_n accepts a language $L \in \mathcal{L}_{0\Sigma}$ then n is an index of L.

A language $L \in \mathcal{L}_{0\Sigma}$ has infinitely many indices (it is accepted by infinitely many TMs).

The theorem of Rice

Preliminaries:

Let M_n be the TM with Gödel number n.

If M_n accepts a language $L \in \mathcal{L}_{0\Sigma}$ then n is an index of L.

A language $L \in \mathcal{L}_{0\Sigma}$ has infinitely many indices (it is accepted by infinitely many TMs).

Index set

Let P be a property of languages of type 0, $\quad P \subseteq \mathcal{L}_{0\Sigma}$

$I(P) = \{n \mid M_n \text{ accepts an } L \in P\}$ is the index set of P.

The theorem of Rice

Informally:
For every non-trivial property P of languages of type 0,
it is undecidable whether the language accepted by a TM has property P.

Every non-trivial property of TMs is undecidable.

The theorem of Rice

Informally:
For every non-trivial property P of languages of type 0,
it is undecidable whether the language accepted by a TM has property P.

Every non-trivial property of TMs is undecidable.

Non-trivial property: Every property P s.t. there is a language of type 0 with property P and not all languages of type 0 have property P.

Note: P is a property of languages, not of Turing machines.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Proof.

Idea: We reduce H_0 (resp. the complement of H_0) to $I(P)$
depending on whether $\emptyset \in P$ or not.

The theorem of Rice

Proof: Case 1: $\emptyset \in P$. We reduce the complement of H_0 to $I(P)$.

We need to construct a TM computable function f such that

$$i \notin H_0 \quad \text{iff} \quad f(i) = j \in I(P)$$

The theorem of Rice

Proof: Case 1: $\emptyset \in P$. We reduce the complement of H_0 to $I(P)$.

We need to construct a TM computable function f such that

$$i \notin H_0 \quad \text{iff} \quad f(i) = j \in I(P)$$

For every i, $f(i) = j$ is constructed as follows:

- Let L be an arbitrary language in $\mathcal{L}_{0\Sigma} \backslash P$ and M_L be a TM which accepts L.
- Let $M_j = M_i^{(2)} M_L^{(1)}$ a 2-tape TM (with Gödel number j) which works as follows:
  - M_j is started with input $\#|^k \underline{\#}$ on tape 1 and $\#\underline{\#}$ on tape 2.
  - M_j works first on tape 2 as M_i (for input 0).
    If M_i halts, M_j then works on tape 1 as M_L.

  Therefore, M_j accepts the language: $L_j = \begin{cases} \emptyset & \text{if } M_i \text{ does not halt on input } 0 \\ L & \text{if } M_i \text{ halts on input } 0. \end{cases}$

The theorem of Rice

Proof: Case 1: $\emptyset \in P$. We reduce the complement of H_0 to $I(P)$.

We need to construct a TM computable function f such that

$$i \notin H_0 \quad \text{iff} \quad f(i) = j \in I(P)$$

For every i, $f(i) = j$ is constructed as follows:

- Let L be an arbitrary language in $\mathcal{L}_{0\Sigma} \backslash P$ and M_L be a TM which accepts L.
- Let $M_j = M_i^{(2)} M_L^{(1)}$ a 2-tape TM (with Gödel number j) which works as follows:
  - M_j is started with input $\#|^k\underline{\#}$ on tape 1 and $\#\underline{\#}$ on tape 2.
  - M_j works first on tape 2 as M_i (for input 0).
    If M_i halts, M_j then works on tape 1 as M_L.

  Therefore, M_j accepts the language: $L_j = \begin{cases} \emptyset & \text{if } M_i \text{ does not halt on input } 0 \\ L & \text{if } M_i \text{ halts on input } 0. \end{cases}$

We know that $\emptyset \in P$ and $L \notin P$. Therefore:

$f(i) = j \in I(P)$ iff $L(M_j) \in P$ iff $L_j = \emptyset$ iff M_i does not halt on 0 iff $i \notin H_0$

Thus, we have reduced the complement of H_0 to $I(P)$.

The theorem of Rice

Proof: Case 2: $\emptyset \notin P$. We reduce H_0 to $I(P)$.

We need to construct a TM computable function f such that

$$i \in H_0 \quad \text{iff} \quad f(i) = j \in I(P)$$

For every i, $f(i) = j$ is constructed as follows:

- $P \neq \emptyset$, so there exists a language $L \in P$, and a TM M_L which accepts L.
- Let $M_j = M_i^{(2)} M_L^{(1)}$ a 2-tape TM (with Gödel number j) which works as follows:
  - M_j is started with input $\#|^k\underline{\#}$ on tape 1 and $\#\underline{\#}$ on tape 2.
  - M_j works first on tape 2 as M_i (for input 0).
    If M_i halts, M_j then works on tape 1 as M_L.

  Therefore, M_j accepts the language: $L_j = \begin{cases} \emptyset & \text{if } M_i \text{ does not halt on input } 0 \\ L & \text{if } M_i \text{ halts on input } 0. \end{cases}$

Since $\emptyset \notin P$ and $L \in P$, we have:

$$f(i) = j \in I(P) \ \text{ iff } \ L_j \in P \ \text{ iff } \ L_j = L \ \text{ iff } \ M_i \text{ halts on } 0 \ \text{ iff } \ i \in H_0$$

Thus, we have reduced H_0 to $I(P)$.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Consequences:

1. The emptiness problem $E = \{n \mid M_n \text{ halts for no input }\}$ is undecidable.

Proof: Take $P = \{\emptyset\}$. The empty language is TM acceptable, i.e. $P \subseteq \mathcal{L}_{0\Sigma}$.
P is non-trivial. Thus. $I(P)$ is undecidable.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Consequences:

1. The emptiness problem $E = \{n \mid M_n \text{ halts for no input }\}$ is undecidable.
2. Let $L \in \mathcal{L}_{0,\Sigma}$. Then $\{n \mid L(M_n) = L\}$ is undecidable.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \dots, M_n, \dots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Consequences:

1. The emptiness problem $E = \{n \mid M_n \text{ halts for no input }\}$ is undecidable.
2. Let $L \in \mathcal{L}_{0,\Sigma}$. Then $\{n \mid L(M_n) = L\}$ is undecidable.

Proof: Take $P = \{L\}$.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Consequences:

1. The emptiness problem $E = \{n \mid M_n \text{ halts for no input }\}$ is undecidable.
2. Let $L \in \mathcal{L}_{0,\Sigma}$. Then $\{n \mid L(M_n) = L\}$ is undecidable.
3. $\{n \mid L(M_n) \text{ is regular }\}$ is undecidable.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Consequences:

1. The emptiness problem $E = \{n \mid M_n \text{ halts for no input }\}$ is undecidable.
2. Let $L \in \mathcal{L}_{0,\Sigma}$. Then $\{n \mid L(M_n) = L\}$ is undecidable.
3. $\{n \mid L(M_n) \text{ is regular }\}$ is undecidable.

Proof: take P to be the set of all regular languages.

The theorem of Rice

Theorem (Henry Gordon Rice, 1953)

Let P be such that $\emptyset \neq P \subsetneq \mathcal{L}_{0,\Sigma}$.

Let $M_0, M_1, \ldots, M_n, \ldots$ be the enumeration of all Turing Machines.

Then $I(P) = \{n \mid M_n \text{ accepts a language } L \in P\}$ is undecidable.

Consequences:

1. The emptiness problem $E = \{n \mid M_n \text{ halts for no input }\}$ is undecidable.
2. Let $L \in \mathcal{L}_{0,\Sigma}$. Then $\{n \mid L(M_n) = L\}$ is undecidable.
3. $\{n \mid L(M_n) \text{ is regular }\}$ is undecidable.
4. $\{n \mid L(M_n) \text{ is context sensitive }\}$ is undecidable.

Decidability and Undecidability results

Logic

- The set of theorems in first-order logic is undecidable

Formal languages

- The Post Correspondence Problem and its consequences

Decidability and Undecidability results

Logic

1. The set of theorems in propositional logic

Decidability and Undecidability results

Logic

1. The set of theorems in propositional logic is decidable

   Idea of the proof: There are sound, complete and terminating algorithms for checking validity of formulae in propositional logic

   (truth tables, resolution, tableaux, DPLL, ...)

Decidability and Undecidability results

Logic

1. The set of theorems in propositional logic is decidable
2. The set of theorems in first-order logic

Decidability and Undecidability results

Logic

1. The set of theorems in propositional logic is decidable
2. The set of theorems in first-order logic is recursively enumerable, but undecidable

Idea of proof:

- For each signature Σ, the set of valid Σ-formulas is recursively enumerable:
  Resolution is a complete deduction system.
- For most signatures Σ, validity is undecidable for Σ-formulas:
  One can easily encode Turing machines in most signatures

Decidability and Undecidability results

Theorem. It is undecidable whether a first order logic formula is valid.

Proof. Suppose there is an algorithm P that, given a first order logic and a formula in that logic, decides whether that formula is valid.

We use P to give a decision algorithm for the language

$\{\langle G(M), w\rangle \,|\, G(M)$ is the Gödel number of a TM M that accepts the string $w\}$.

As the latter problem is undecidable this will show that P cannot exist.

Given M and w, we create a FOL signature by declaring

- a constant ϵ,
- a unary function symbol a for every letter a in the alphabet, and
- a binary predicate q for every state q of M.

Decidability and Undecidability results

Proof (ctd.)

Consider the following interpretation of this logic:

- Variables x range over strings over the given alphabet,
- ϵ denotes the empty string,
- $a(w)$ denotes the string aw, and
- $q(x, y)$ indicates that M, when given input w, can reach a configuration with state q, in which xy is on the tape, with x in reverse order, and the head of M points at the first position of y.

Under this interpretation $s(\epsilon, w)$ is certainly a true formula, as the initial configuration is surely reachable (where w is a representation of w made from the constant and function symbols of the logic).

Furthermore the formula $\exists x \exists y : h(x, y)$ holds iff M accepts w.

Decidability and Undecidability results

Proof (ctd.) Whenever M has a transition from state q to state r, reading a, writing b, and moving right, the formula

$$\forall x \forall y : q(x, ay) \rightarrow r(bx, y)$$

holds. Likewise, if M has a transition from state q to state r, reading a, writing b, and moving left, the formulas

$$\forall x \forall y : q(cx, ay) \rightarrow r(x, cby)$$

hold for every choice of a letter c. In addition we have

$$\forall x \forall y : q(\epsilon, ay) \rightarrow r(\epsilon, by),$$

covering the case that M cannot move left, because its head is already in the left-most position.

Decidability and Undecidability results

Proof (ctd.)

Finally, there are variants of the formulas above for the case that a is the blank symbol and that square of the tape is visited for the first time:

$$\begin{aligned}
\forall x \forall y : q(x, \epsilon) &\quad\rightarrow\quad r(bx, \epsilon)\\
\forall x \forall y : q(cx, \epsilon) &\quad\rightarrow\quad r(x, cb)\\
\forall x \forall y : q(\epsilon, \epsilon) &\quad\rightarrow\quad r(\epsilon, b).
\end{aligned}$$

Let T be the conjunction of all implication formulas mentioned above. As M has finitely many transitions and the alphabet is finite, this conjunction is finite as well, and thus a formula of first order logic.

Decidability and Undecidability results

Proof (ctd.) Now consider the formula

$$s(\epsilon, w) \wedge T \rightarrow \exists x \exists y : h(x, y).$$

In case M accepts w, there is a valid computation leading to an accept state. Each step therein corresponds with a substitution instance of one of the conjuncts in T, and using the laws of first order logic it is easy to check that the formula above is provable and thus true under all interpretations.

If, on the other hand, the formula above is true under all interpretations, it is surely true in the given interpretation, which implies that M has an accepting computation starting on w.

Thus, in order to decide whether or not M accepts w, it suffices to check whether or not the formula above is a theorem of first order logic.

Decidability and Undecidability results

Logic

- The set of theorems in first-order logic is undecidable

Formal languages next time

- The Post Correspondence Problem and its consequences