# Advanced Topics in Theoretical Computer Science

## Part 4: Computability and (Un-)Decidability

10.01.2013

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

# Contents

- 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

# Contents

- 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

# The Church-Turing Thesis

**Informally:** The functions which are intuitively computable are exactly the functions which are Turing computable.

# The Church-Turing Thesis

**Informally:** The functions which are intuitively computable are exactly the functions which are Turing computable.

**Instances of this thesis:** all known models of computation

- Turing machines
- Recursive functions
- $\lambda$-functions
- all known programming languages (imperative, functional, logic)

provide the same notion of computability

# Alonzo Church

**Alonzo Church** (1903-1995)

- studied in Princeton; PhD in Princeton
- Postdoc in Göttingen
- Professor: Princeton and UCLA
- Layed the foundations of theoretical computer science (e.g. introduced the $\lambda$-calculus)
- One of the most important computer scientists

# Alonzo Church

**PhD Students:**

- **Peter Andrews**: automated reasoning
- **Martin Davis**: Davis-Putnam procedure (automated reasoning)
- **Leon Henkin**: (Standard) proof of completeness of predicate logic
- **Stephen Kleene**: Regular expressions
- **Dana Scott**: Denotational Semantics, Automata theory
- **Raymond Smullyan**: Tableau calculi
- **Alan Turing**: Turing machines, Undecidability of the halting problem
- ... and many others

# Contents

# Computability and (Un-)decidability

Known undecidable problems (Theoretical Computer Science I)

- The halting problem for Turing machines
- Equivalence problem

# Computability and (Un-)decidability

Known undecidable problems (Theoretical Computer Science I)

- The halting problem for Turing machines
- The equivalence problem

**Consequences:**

- All problems about programs (TM) which are non-trivial (in a certain sense) are undecidable (Theorem of Rice)
- Identify undecidable problems outside the world of Turing machines
  - Validity/Satisfiability in First-Order Logic
  - The Post Correspondence Problem

# Computability and (Un-)decidability

Known undecidable problems (Theoretical Computer Science I)

- The halting problem for Turing machines
- The equivalence problem

**Consequences:**

- All problems about programs (TM) which are non-trivial (in a certain sense) are undecidable (Theorem of Rice)
- Identify undecidable problems outside the world of Turing machines
  - Validity/Satisfiability in First-Order Logic
  - The Post Correspondence Problem

# Computability and (Un-)decidability

**The Theorem of Rice** (informally)

Variant 1

For each non-trivial property $P$ of (partial) functions:

It is undecidable, whether the function computed by a Turing machine has property $P$.

Variant 2

Every non-trivial property $P$ of Turing Machines is undecidable.

# Computability and (Un-)decidability

**The Theorem of Rice** (informally)

Variant 1

For each non-trivial property $P$ of (partial) functions:

It is undecidable, whether the function computed by a Turing machine has property $P$.

Variant 2

For each non-trivial property $P$ of languages of type 0:

It is undecidable, whether the language accepted by a Turing machine has property $P$.

# Computability and (Un-)decidability

**The Theorem of Rice** Variant 1

For each non-trivial property $P$ of (partial) functions:

It is undecidable, whether the function computed by a Turing machine has property $P$.

Generalization of Variant 1:

The same holds for other computability models:

- algorithms
- Java programs
- $\lambda$ expressions
- recursive functions
- etc.

# The theorem of Rice

**Examples of non-trivial properties of functions:**

- Monotonicity: $\{f \mid f(i) \leq f(i+1) \text{ for all } i\}$
- Equivalence: $\{f \mid f = g\}$ for a given $g$
- Image: $\{f \mid j \in \mathit{Im}(f)\}$ for a given $j$
- Square function $\{f \mid f(i) = i^2 \text{ for all } i\}$

# Plan

- Recall: Acceptance and Decidability
- Recall: Undecidability results
  The halting problem
  Undecidability proofs via reduction
- The theorem of Rice (Variant 2)

# 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 $\Sigma_0$ 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 $\Sigma_0$ with $\#, Y, N \not\in \Sigma_0$; $\mathcal{M}=(K, \Sigma, \delta, s)$ 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 $\Sigma_0$ 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 an instance of $L_1$ into an instance 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.** In the lecture from 13.12.2012 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 $\Sigma$ then $\langle w \rangle_I = \langle i_1, \ldots, i_n \rangle = \prod_{j=1}^{n} p(j)^{i_j}$

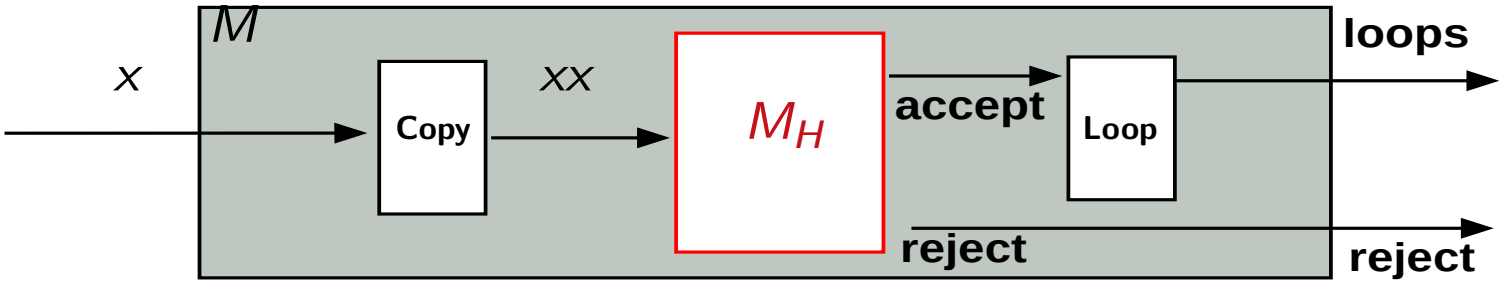
This shows, in particular, that we can represent w.l.o.g. words as natural numbers and languages as sets of natural numbers.

# Undecidability of the halting problem

**Proposition:** $HALT = \{G(\mathcal{M})w \mid \mathcal{M} \text{ halts on input } w\}$ is not decidable.

Proof: Assume, in order to derive a contradiction, that there exists a TM $M_H$ which halts on every input and accepts only inputs in $HALT$.

We construct the following TM:



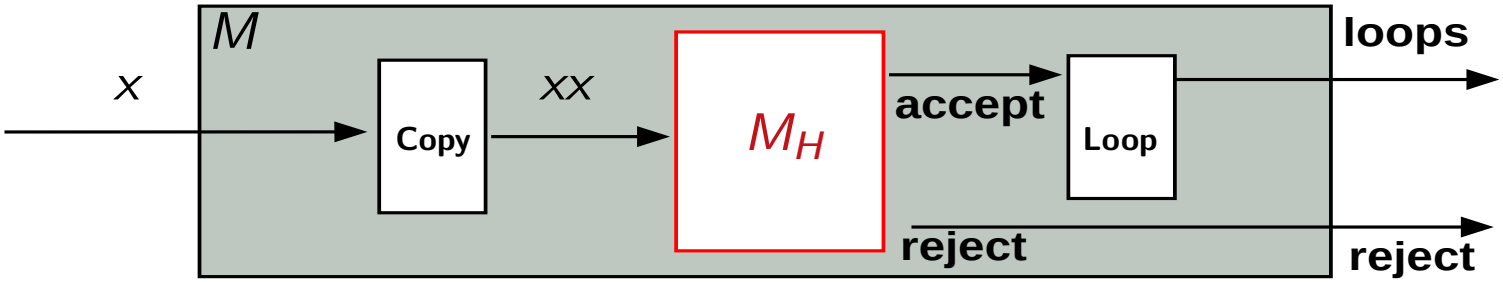

1. Let $x$ be the input.
2. Copy the input. Let $xx$ be the result.
3. Decide using $M_H$ if $xx \in HALT$
4. If yes: write infinitely many 1s to the right.
5. If no: halt

# Undecidability of the halting problem

**Proposition:** $HALT = \{G(\mathcal{M})w \mid \mathcal{M} \text{ halts on input } w\}$ is not decidable.

Proof: Assume, in order to derive a contradiction, that there exists a TM $M_H$ which halts on every input and accepts only inputs in $HALT$.

What happens when we start $M$ with input $G(M)$?




Case 1: $M$ started with $G(M)$ halts: Then $G(M)G(M) \notin HALT$ Contradiction!

Case 2: $M$ started with $G(M)$ does not halt: Then $G(M)G(M) \in HALT$ Contradiction!

# Undecidability proofs: Example

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

Proof: Similar to the proof of the halting problem.

Exercise

# Undecidability proofs: Example

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

Proof: Similar to the proof of the halting problem.

**Reformulation** using numbers instead of words:

Gödelization $\mapsto$ Gödel numbers

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

$M_n$ is the TM with Gödel number $n$.

$$K = \{n \mid M_n \text{ halts on input } n\}$$

# 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 $\quad (M_i \text{ halts for input } i \text{ iff } M_j \text{ halts for input } 0)$.

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

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

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).

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 $I(P)$.

Let $L$ be an arbitrary language in $\mathcal{L}_{0\Sigma} \setminus P$ and $M_L$ be a TM which accepts $L$.

From $M_L$ we construct a 2-tape TM, $M_j = M_i^{(2)} M_L^{(1)}$ with Gödel number $f(i) = j$ 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 $0 \in P$ and $L \notin P$. Therefore:

$f(i) = j \in I(P)$ iff $L_j \in P$ 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 $I(P)$.

$P \neq \emptyset$, so there exists a language $L \in P$, and a TM $M_L$ which accepts $L$.

For every $i \in \mathbb{N}$, let $M_j = M_i^{(2)} M_L^{(1)}$ a 2-tape TM 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:

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

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:** 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.

# 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 $\Sigma$, the set of valid $\Sigma$-formulas is recursively enumerable:
  Resolution is a complete deduction system.
- For most signatures $\Sigma$, validity is undecidable for $\Sigma$-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

$\{G(M)w | G(M)$ is the Gödelisation 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 $\epsilon$,
- 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,
- $\epsilon$ 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:

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

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.