Part 4: Computability and (Un-)Decidability (2)

15.01.2015

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de
Last time

Theorem of Rice:

- All problems about programs (TM) which are non-trivial (in a certain sense) are undecidable

Today

Identify undecidable problems outside the world of Turing machines

- Validity/Satisfiability in First-Order Logic
- The Post Correspondence Problem
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 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 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

\[ \{ \langle G(M), w \rangle \mid G(M) \text{ is the Gödel number of a TM } M \text{ that accepts the string } w \} \].

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:

\[
\forall x \forall y : q(x, \epsilon) \rightarrow r(bx, \epsilon)
\]
\[
\forall x \forall y : q(cx, \epsilon) \rightarrow r(x, cb)
\]
\[
\forall x \forall y : q(\epsilon, \epsilon) \rightarrow r(\epsilon, b).
\]

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

\[ s(\epsilon, w) \land T \rightarrow \exists x \exists y : h(x, y). \]

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

• The Post Correspondence Problem and its consequences
Decidability and Undecidability results

Formal languages

• The Post Correspondence Problem and its consequences
Post Correspondence Problem

Idea: We consider non-empty strings over the alphabet \( \{a, b\} \).

For example “aaabba”.

Assume that we have \( n \) pairs of strings \((x_1, y_1), \ldots, (x_n, y_n)\).

Post correspondence problem:
Determine whether there is a set of indices \( i_1, \ldots, i_m \) such that

\[
x_{i_1}x_{i_2}\ldots x_{i_m} = y_{i_1}y_{i_2}\ldots y_{i_m}.
\]

This can contain repeated indices, miss certain indices, \ldots
Definition

A correspondence system (CS) $P$ is a finite rule set over an alphabet $\Sigma$.

$$P = \{(p_1, q_1), \ldots, (p_n, q_n)\} \text{ with } p_i, q_i \in \Sigma^*$$

An index sequence $I = i_1 \ldots i_m$ of $P$ is a sequence with $1 \leq i_k \leq n$ for all $k$. For every index sequence $I$ we denote $p_I = p_{i_1} \ldots p_{i_m}$ and $q_I = q_{i_1} \ldots q_{i_m}$.

A partial solution is an index set $I$ such that $p_I$ is a prefix of $q_I$ or $q_I$ is a prefix of $p_I$.

A solution is an index set $I$ such that $p_I = q_I$.

A (partial) solution with given start is a (partial) solution in which the first index $i_1$ is given.

The Post correspondence problem (PCP) is the question whether a given correspondence system $P$ has a solution.
Example:

Let $P = \{(a, ab), (b, ca), (ca, a), (abc, c)\}$.

- $I = 1, 2, 3, 1, 4$ is a solution:
  
  \[
  p_I = p_1 p_2 p_3 p_1 p_4 = a \ b \ c \ a \ a \ b \ c = abcaabca = \\
  ab \ c \ a \ a \ b \ c = q_1 q_2 q_3 q_1 q_4 = q_I
  \]

- $J = 1, 2, 3$ is a partial solution:
  
  \[
  p_J = p_1 p_2 p_3 = abca \text{ is a prefix of } q_J = abcaa
  \]

- There are no solutions with given start 2, 3 or 4.
Plan

We will show that the Post correspondence problem is undecidable.

The proof consists of the following steps:

• We identify two types of “rewrite” systems
  Semi-Thue systems (STS) and Post Normal Systems (PNS).
• We show that the TM computable functions are also STS/PNS computable.
• We define $\text{Trans}_G = \{(v, w) \mid v \Rightarrow^* w, v, w \in \Sigma^+\}$ and show that there exist STS/PNS $G$ such that $\text{Trans}_G$ is undecidable.
• We assume (to derive a contradiction) that a version of the Post correspondence problem is decidable and show that then also $\text{Trans}_G$ is decidable (which is clearly impossible).
**Set of rules.** A set of rules over an alphabet $\Sigma$ is a finite subset $R \subseteq \Sigma^* \times \Sigma^*$. We also write $u \rightarrow_R v$ for $(u, v) \in R$.

$R$ is $\epsilon$-free if for all $(u, v) \in R$ we have $u \neq \epsilon$ and $v \neq \epsilon$. 

**STS and PNS**

**Set of rules.** A set of rules over an alphabet $\Sigma$ is a finite subset $R \subseteq \Sigma^* \times \Sigma^*$. We also write $u \rightarrow_R v$ for $(u, v) \in R$.

$R$ is $\epsilon$-free if for all $(u, v) \in R$ we have $u \neq \epsilon$ and $v \neq \epsilon$.

**Semi-Thue System.** In a semi-Thue System, a word $w$ is transformed in a word $w'$ by applying one of the rules $(u, v)$ in $R$.

---

**Definition.** A semi-Thue System (STS) is a pair $G = (\Sigma, R)$ consisting of an alphabet $\Sigma$ and a set of rules $R$. $G$ is $\epsilon$-free if $R$ is $\epsilon$-free.

$w \Rightarrow_G w'$ iff $\exists u \rightarrow_R v, \exists w_1, w_2 \in \Sigma^* (w = w_1uw_2$ and $w' = w_1vw_2)$
Let \( G \) be the following semi-Thue system:

\[
G = (\{a, b\}, \{ab \rightarrow bba, ba \rightarrow aba\})
\]

\[
ababa \Rightarrow bbaaba \Rightarrow bbabbaa
\]

\[
ababa \Rightarrow aababa \Rightarrow aabbbbaa.
\]

The rule application in not deterministic.
Definition. A Post Normal System (PNS) is a pair $G = (\Sigma, R)$ where $\Sigma$ is an alphabet and a set of rules $R$. $G$ is $\epsilon$-free if $R$ is $\epsilon$-free.

It differs from a semi-Thue system in the way $\Rightarrow_G$ is defined:

$$w \Rightarrow_G w' \iff \exists u \rightarrow_R v, \exists w_1 \in \Sigma^* (w = uw_1 \text{ and } w' = w_1v)$$

Definition. A computation in a STS or a PNS $G$ is a sequence $w_1, \ldots, w_n$ with $w_i \Rightarrow_G w_{i+1}$ for all $i \in \{1, \ldots, n-1\}$.

The computation does not continue if there exists no $w_{n+1}$ with $w_n \Rightarrow_G w_{n+1}$.

If there exists $n \geq 1$ with $w_1 \Rightarrow_G \cdots \Rightarrow_G w_n$ we write: $w_1 \Rightarrow_G^* w_n$. 
Example

Let $G$ be the following Post Normal System:

$$G = (\{a, b\}, \{ab \rightarrow bba, ba \rightarrow aba, a \rightarrow ba\})$$

Then:

- $ababa \Rightarrow ababba \Rightarrow babbaba \Rightarrow bbabaaba$

- $ababa \Rightarrow bababa \Rightarrow babaaba \Rightarrow baabaaba \Rightarrow abaabaaba \Rightarrow \ldots$

(infinite computation)
Definition. A partial function $f : \Sigma_1^* \rightarrow \Sigma_2^*$ is **STS computable** (PNS-computable) iff there exists a **STS** (a **PNS**) $G$ s.t. for all $w \in \Sigma_1^*$

- $\forall u \in \Sigma_2^*, \ [w] \xrightarrow{G}^* [u] \iff f(w) = u$
- $\exists v \in \Sigma_2^*, \ [w] \xrightarrow{G}^* [v] \iff f(w) \text{ undefined.}$

**Note:** $[,], \rangle$ are special symbols

$F_{\text{STS}}^{\text{part}}$ : the family of all (partial) STS computable functions

$F_{\text{PNS}}^{\text{part}}$ : the family of all (partial) PNS computable functions
Post Correspondence Problem

**Theorem** \( T_{\text{part}}^{\text{TM}} \subseteq F_{\text{part}}^{\text{STS}}; \ T_{\text{part}}^{\text{TM}} \subseteq F_{\text{part}}^{\text{PNS}}. \)

**Proof:**

**Idea:** show that we can simulate the way a TM works using a suitable STS. We then show that we can slightly change the STS and obtain a PNS which simulates the TM.

From the proof it can be seen that we can simulate any TM using a \( \epsilon \)-free STS and \( \epsilon \)-free PNS.

The full proof is rather long and is not presented here. It can be found on pages 309-311 in the book “Theoretische Informatik” (3. Auflage) by Erk and Priese.
Post Correspondence Problem

\[ Trans_G = \{(v, w) \mid v \Rightarrow^*_G w \land v, w \in \Sigma^+\} \]

**Theorem.**

There exists an \( \epsilon \)-free STS \( G \) such that \( Trans_G \) is undecidable.

There exists an \( \epsilon \)-free PNS \( G \) such that \( Trans_G \) is undecidable.

**Proof.**

We can reduce \( K = \{n \mid M_n \text{ halts on input } n\} \) to \( Trans_G \) for a certain STS (PNS) \( G \).

Let \( G \) be an \( \epsilon \)-free STS or PNS which computes the function of the TM

\[ M = M_K M_{\text{delete}} \]

where \( M_K \) is the TM which accepts \( K \) and \( M_{\text{delete}} \) deletes the band after \( M_K \) halts (such a TM can easily be constructed because \( M_K = M_{\text{prep}} U_0 \); the halting configurations of the universal TM \( U_0 \) are of the form \( h_U, \#|n\#|m\#\)).

Input \( v \): \( M_K \) halts iff \( M_v \) halts on \( v \). If \( M_K \) halts, \( M_{\text{delete}} \) deletes the tape.
Post Correspondence Problem

Proof. (ctd.)

Assume \( Trans_G \) decidable. We show how to use \( G \) and the decision procedure for \( Trans_G \) to decide \( K \):

For \( v = [| \ldots |] \) and \( w = [\epsilon] \) we have:

\[
(v, w) \in Trans_G \quad \text{iff} \quad (v \Rightarrow^*_G w)
\]

\[
\text{iff} \quad M = M_K M_{\text{delete}} \text{ halts for input } |n \text{ with } \#
\]

\[
\text{iff} \quad M_K \text{ halts for input } |n
\]

\[
\text{iff} \quad n \in K.
\]
Post Correspondence Problem

**Theorem** For every $\varepsilon$-free semi-Thue System $G$ and every pair of words $w', w'' \in \Sigma^+$ there exists a Post Correspondence System $P_{G,w',w''}$ such that

$$P_{G,w',w''} \text{ has a solution with given start } \iff w' \Rightarrow^*_G w'' .$$

**Proof:** Assume that we are given

- $G$ an $\varepsilon$-free STS $G = (\Sigma, R)$ with $|\Sigma| = m$ and $R = \{u_1 \to v_1, \ldots, u_n \to v_n\}$ with $u_i, v_i \in \Sigma^+$
- $w', w'' \in \Sigma^+$

We construct the correspondence system $P_{G,w',w''} = \{(p_i, q_i) \mid 1 \leq i \leq k\}$ with $k = n + m + 3$ over the alphabet $\Sigma_X = \Sigma \cup X$ with:

- the first $n$ rules are the rules in $R$
- the rule $n + 1$ is $(X, Xw'X)$; the rule $n + 2$ is $(w''XX, X)$
- the rules $n + 2 + 1, \ldots, n + 2 + m$ are $(a, a)$ for every $a \in \Sigma$
- the last rule is $(X, X)$
- the index for the given start is $n + 1$. 
Example

\[ G = (\Sigma, R) \text{ with } \Sigma = \{a, b, c\} \text{ and } R = \{ca \rightarrow ab, ab \rightarrow c, ba \rightarrow a\}. \]

For the word pair \( w' = caaba, w'' = abc \) we have

\[ w' = caaba \Rightarrow_2 caca \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w'' \]

\[ P_{G,w',w''} = \{(ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X), (a, a), (b, b), (c, c), (X, X)\} \]

We can see that \( P_{G,w',w''} \) has a solution with start \( n + 1 \) iff \( w' \Rightarrow^*_G w'' \)

\[ p_4 \quad X \quad = XcaabaX \quad = q_4 \]
Example

\[ G = (\Sigma, R) \text{ with } \Sigma = \{a, b, c\} \text{ and } R = \{ca \rightarrow ab, ab \rightarrow c, ba \rightarrow a\}. \]

For the word pair \( w' = caaba, w'' = abc \) we have

\[
w' = caaba \Rightarrow_2 caac \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w''
\]

\[ P_{G, w', w''} = \{ (ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X), (a, a), (b, b), (c, c), (X, X) \} \]

We can see that \( P_{G, w', w''} \) has a solution with start \( n + 1 \) iff \( w' \Rightarrow^*_G w'' \)

\[
p_{486} = Xca = XcaabaXca = q_{486}
\]
Example

\[ G = (\Sigma, R) \text{ with } \Sigma = \{a, b, c\} \text{ and } R = \{ca \rightarrow ab, ab \rightarrow c, ba \rightarrow a\}. \]

For the word pair \( w' = caaba, w'' = abc \) we have

\[ w' = caaba \Rightarrow_2 caca \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w'' \]

\[ P_{G,w',w''} = \{(ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X), (a, a), (b, b), (c, c), (X, X)\} \]

We can see that \( P_{G,w',w''} \) has a solution with start \( n + 1 \) iff \( w' \Rightarrow^*_G w'' \)

\[ p_{4862} = Xcaab \] \[ = XcaabaXcac \] \[ = q_{4862} \]
Example

\[ G = (\Sigma, R) \text{ with } \Sigma = \{a, b, c\} \text{ and } R = \{ca \rightarrow ab, ab \rightarrow c, ba \rightarrow a\}. \]

For the word pair \( w' = caaba, w'' = abc \) we have

\[
w' = caaba \Rightarrow_2 caca \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w''
\]

\[
P_{G, w', w''} = \{(ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X), (a, a), (b, b), (c, c), (X, X)\}
\]

We can see that \( P_{G, w', w''} \) has a solution with start \( n + 1 \) iff \( w' \Rightarrow^*_G w'' \)

\[
p_{486269} = XcaabaX = XcaabaXcacaX = q_{486269}
\]
Example

\[ G = (\Sigma, R) \] with \( \Sigma = \{a, b, c\} \) and \( R = \{ca \to ab, ab \to c, ba \to a\} \).

For the word pair \( w' = caaba, w'' = abc \) we have

\[
w' = caaba \Rightarrow_2 caca \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w''
\]

\[ P_{G,w',w''} = \{(ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X), (a, a), (b, b), (c, c), (X, X)\} \]

We can see that \( P_{G,w',w''} \) has a solution with start \( n + 1 \) iff \( w' \Rightarrow^*_G w'' \)

\[
p_{48626986} = XcaabaXca = XcaabaXcacaXca = q_{48626986}
\]
Example

\( G = (\Sigma, R) \) with \( \Sigma = \{a, b, c\} \) and \( R = \{ca \rightarrow ab, ab \rightarrow c, ba \rightarrow a\} \).

For the word pair \( w' = caaba, w'' = abc \) we have

\[
\begin{align*}
w' &= caaba \Rightarrow_2 cac \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w'' \\
P_{G, w', w''} &= \{(ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X), (a, a), (b, b), (c, c), (X, X)\} \\
We can see that \( P_{G, w', w''} \) has a solution with start \( n + 1 \) iff \( w' \Rightarrow^*_G w'' \)
\]

\( p_{4862698619} = XcaabaXcacaX = XcaabaXcacaXcaabX = q_{4862698619} \)
Example

$G = (\Sigma, R)$ with $\Sigma = \{a, b, c\}$ and $R = \{ca \rightarrow ab, ab \rightarrow c, ba \rightarrow a\}$.

For the word pair $w' = caaba, w'' = abc$ we have

$$w' = caaba \Rightarrow_2 caca \Rightarrow_1 caab \Rightarrow_2 cac \Rightarrow_1 abc = w''$$

$P_{G,w',w''} = \{(ca, ab), (ab, c), (ba, a), (X, XcaabaX), (abcXX, X)\}$

We can see that $P_{G,w',w''}$ has a solution with start $n + 1$ iff $w' \Rightarrow^* w''$

$p_{4862698619} = XcaabaXcacaXcacaXcaabX = q_{4862698619}$

The successive application of rules 2, 1, 2, 1 corresponds to the solution

$l = 4, 8, 6, 2, 6, 9, 8, 6, 1, 9, 8, 6, 2, 9, 1, 8, 9, 5$

4,4: begin/end; Underlines: rule applications. Remaining numbers: copy symbols such that rule applications at the desired position. $X$ separates the words in $G$-derivations.

$p_l = XcaabaXcacaXcaabXcacXabcXX = q_l$
Theorem  For every $\epsilon$-free semi-Thue System $G$ and every pair of words $w', w'' \in \Sigma^+$ there exists a Post Correspondence System $P_{G,w',w''}$ such that

$$P_{G,w',w''}$$ has a solution with given start iff $w' \Rightarrow^* G w''$.

Proof: Assume that we are given

- $G$ an $\epsilon$-free STS $G = (\Sigma, R)$ with $|\Sigma| = m$ and $R = \{u_1 \rightarrow v_1, \ldots, u_n \rightarrow v_n\}$ with $u_i, v_i \in \Sigma^+$
- $w', w'' \in \Sigma^+$

We construct the correspondence system $P_{G,w',w''} = \{(p_i, q_i) \mid 1 \leq i \leq k\}$ with $k = n + m + 3$ over the alphabet $\Sigma_X = \Sigma \cup X$ with:

- the first $n$ rules are the rules in $R$
- the rule $n + 1$ is $(X, Xw'X)$; the rule $n + 2$ is $(w''XX, X)$
- the rules $n + 2 + 1, \ldots, n + 2 + m$ are $(a, a)$ for every $a \in \Sigma$
- the last rule is $(X, X)$
- the index for the given start is $n + 1$. 
Proof (ctd.) We show that $P_{G,w',w''}$ has a solution iff $w \Rightarrow^*_G w''$.

Occurrences of $X \mapsto \to$ in the solution index $n + 2$ must occur.

Assume $(n + 1)l'(n + 2)l''$ is a solution in which $l'$ does not contain $n + 1$, nor $n + 2$. By careful analysis of the equality $p_{(n+1)}l'(n+2)l'' = q_{(n+1)}l'(n+2)l''$ we note the following:

1. no $XX$ in $p_{(n+1)}l', q_{(n+1)}l'$;
2. $p_{(n+1)}l'(n+2)$, and $q_{(n+1)}l'(n+2)$ end on $XX$;
3. $p_{(n+1)}l'(n+2)l'' = Xp \, w''XXp \, l'' = Xw'Xq \, l'Xq \, l''$, so:
   - $l'$ starts with $l_1, (n + m + 3)$ with $p_{l_1(n+m+3)} = w'X$.
   - Then $q_{l_1,n+m+3} = w_2X$ for some $w_2 \neq \epsilon$.
   - $l_1$ contains only indices in $\{1, \ldots, n\} \cup \{n + 3, \ldots, n + 2 + m\}$.
   - Therefore, $w' \Rightarrow^*_G w_2$. 
Post Correspondence Problem

Proof (ctd.)

From (1) and (2) it follows that $p(n+1)l'(n+2) = q(n+1)l'(n+2)$.

Thus, if $P_{G,w',w''}$ has a solution then it has a solution of the form $(n+1)l'(n+2)$, such that $l'$ does not contain $(n+1)$ or $(n+2)$.

From (3), by induction, we can show that

$$l' = l_1, (n + m + 3), l_2, (n + m + 3), \ldots, l_k, (n + m + 3),$$

where $l_j$ contains only indices in $\{1, \ldots, n\} \cup \{n + 3, \ldots, n + 2 + m\}$.

Then $p_{l'} = w'Xw_2X \ldots Xw_{l-1}X$ and $q_{l'} = w_2X \ldots Xw_lX$

for words $w_2, \ldots, w_l$ with

$$w' \Rightarrow^* G w_2 \Rightarrow^* G \cdots \Rightarrow^* G w_l$$
Post Correspondence Problem

Proof (ctd.)

Thus, for every solution \( I = (n + 1)I' (n + 2) \) we have:

\[
p_I = Xw'Xw_2 \ldots Xw_{l-1}Xw''XX = q_I
\]

with \( w' \Rightarrow_G^* w_2 \Rightarrow_G^* \cdots \Rightarrow_G^* w_l = w'' \).

Conversely, one can prove by induction that if \( w' = w_1 \Rightarrow_G^* w_2 \Rightarrow_G^* \cdots \Rightarrow_G^* w_l = w'' \) is a computation in \( G \) then there exists a partial solution \( I \) of \( P_{G,w',w''} \) with given start \( n + 1 \) and

\[
p_I = Xw'Xw_2 \ldots Xw_{l-1}X \quad q_I = Xw'Xw_2 \ldots Xw_{l-1}Xw_lX
\]

Then \( I, (n + 2) \) is a solution if \( w_l = w'' \).
Theorem. Assume $|\Sigma| \geq 2$. The Post Correspondence Problem is undecidable.

Proof:
1. We first show that PCP with given start is undecidable.

   Assume that the PCP with given start is decidable. By the previous result it would follow that $\text{Trans}_G$ is decidable for every $\epsilon$-free STS $G$. We showed that there exists at least one $\epsilon$-free STS $G$ for which $\text{Trans}_G$ is undecidable. Contradiction. Thus, the PCP with given start is undecidable.

2. We prove that PCP is undecidable.

   For this, we show that for every PCP $P = \{(p_i, q_i) \mid 1 \leq i \leq n\}$ with given start $j_0$ we can construct a PCP $P'$ such that $P$ has a solution iff $P'$ has a solution.

   Construction: New symbols $X, Y$; two types of encodings of words:
   
   $w = c_1 \ldots c_n \mapsto \overline{w} = Xc_1Xc_2 \ldots Xc_n$; \hspace{1cm} $\overline{w} = c_1Xc_2 \ldots Xc_nX$
   
   $P' = \{(\overline{p_1}, \overline{q_1}), \ldots, (\overline{p_n}, \overline{q_n}), (\overline{p_{j_0}}, X\overline{q_{j_0}}), (XY, Y)\}$

   A solution of $P'$ can only start with rule $(n + 1)$ (only rule where both sides start with same symbol). $P$ has solution with start $j_0$ iff $P'$ has a solution.
Theorem It is undecidable whether a context free grammar is ambiguous.

Proof. Assume that the problem is decidable. Construct algorithm for solving the PCP.

Let $T = \{(u_1, v_1), \ldots, (u_n, v_n)\}$ a CS over $\Sigma_1$; $\Sigma' = \Sigma_1 \cup \{a_1, \ldots, a_n\}$.

$L_{T,1} = \{a_{im} \ldots a_{i1} u_{i1} \ldots u_{im} | m \geq 1, 1 \leq i \leq n\}$ generated by c.f. grammar $G_{T,1}$.

$G_{T,1} = (\{S_1\}, \Sigma', R_1, S_1), R_1 = \{S_1 \rightarrow a_i S_1 u_i | 1 \leq i \leq n\} \cup \{S_1 \rightarrow a_i u_i\}$

$L_{T,2} = \{a_{im} \ldots a_{i1} v_{i1} \ldots v_{im} | m \geq 1, 1 \leq i \leq n\}$ generated by c.f. grammar $G_{T,2}$.

$G_{T,2} = (\{S_2\}, \Sigma', R_2, S_2), R_2 = \{S_2 \rightarrow a_i S_2 v_i | 1 \leq i \leq n\} \cup \{S_2 \rightarrow a_i v_i\}$

$G_{T,1}, G_{T,2}$ are unambiguous. Let $G_T = (\{S, S_1, S_2\}, \Sigma', R_1 \cup R_2 \cup \{S \rightarrow S_1, S \rightarrow S_2\}, S)$.

$T$ has a solution iff $\exists w \in L_{T,1} \cap L_{T,2}$

iff $\exists w \in L(G)$ with two different derivations iff $G_T$ ambiguous.
Theorem: It is undecidable whether the intersection of two
- deterministic context-free languages (DCFL)
- non-ambiguous context-free languages
- context-free languages
is empty.

Proof. Assume that one of the problems is decidable.

Let $T = \{(u_1, v_1), \ldots, (u_n, v_n)\}$ a CS over $\Sigma$; $\Sigma' = \Sigma \cup \{a_1, \ldots, a_n\}, c \notin \Sigma'$.

$L_1 = \{wcw^R \mid w \in (\Sigma')^*\}$: non-ambiguous, deterministic.
$L_2 = \{u_{i_1} \ldots u_{i_m} a_{i_m} \ldots a_{i_1} c a_{j_1} \ldots a_{j_l} v_{j_l}^R \ldots v_{j_1}^R \mid m, l \geq 1, i_k, j_p \in \{1, \ldots, n\}\}$
$L_2$ non-ambiguous, deterministic (see proof in the book by Erk and Priese)

$T$ has a solution iff $\exists k \geq 1 \exists i_1, \ldots, i_k : u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$
iff $\exists k \geq 1 \exists i_1, \ldots, i_k : u_{i_1} \ldots u_{i_k} a_{i_k} \ldots a_{i_1} = (a_{i_1} \ldots a_{i_k} v_{i_1}^R \ldots v_{i_k}^R)^R$
iff $\exists x \in L_2$ such that $x = wcw^R$ iff $\exists x \in L_2 \cap L_1$

If we can always decide whether $L_1 \cap L_2 = \emptyset$ then PCP decidable!
Theorem It is undecidable whether for a context free language \( L \subseteq \Sigma^* \) with \( |\Sigma| > 1 \) we have \( L = \Sigma^* \).

Proof. Assume that is was decidable whether \( L = \Sigma^* \). We show that then it would be decidable whether \( L_1 \cap L_2 = \emptyset \) for DCFL.

Let \( L_1, L_2 \) DCFL languages over \( \Sigma \). Then \( L_1 \cap L_2 = \emptyset \) iff \( \overline{L_1 \cup \overline{L_2}} = \Sigma^* \) iff \( \overline{L_1 \cup \overline{L_2}} = \Sigma^* \).

Note that DCFL’s are closed under complement. Then \( \overline{L_1}, \overline{L_2} \in \mathcal{L}_2 \), so \( \overline{L_1 \cup \overline{L_2}} \in \mathcal{L}_2 \).

Then we could use the decision procedure to check whether \( \overline{L_1 \cup \overline{L_2}} = \Sigma^* \), i.e. to check whether \( L_1 \cap L_2 = \emptyset \). This is a contradiction, since we proved that it is undecidable whether the intersection of two DCFLs is empty.
Undecidable problems in formal languages

**Theorem** The following problems are undecidable for context-free languages $L_1$, $L_2$ and regular languages $R$ over every alphabet $\Sigma$ with at least two elements.

1. $L_1 = L_2$
2. $L_2 \subseteq L_1$
3. $L_1 = R$
4. $R \subseteq L_1$

**Proof:** Let $L_1$ be an arbitrary context-free language. Choose $L_2 = \Sigma_2^*$. Then $L_2$ is regular and:

- $L_1 = L_2$ iff $L_1 = \Sigma^*$ (1 and 3)
- $L_2 \subseteq L_1$ iff $L_1 = \Sigma^*$ (2 and 3)
### Undecidable problems for $L_2$

<table>
<thead>
<tr>
<th>decidable</th>
<th>undecidable</th>
</tr>
</thead>
<tbody>
<tr>
<td>$w \in L(G)$</td>
<td>$G$ ambiguous</td>
</tr>
<tr>
<td>$L(G) = \emptyset$</td>
<td>$D_1 \cap D_2 = \emptyset$</td>
</tr>
<tr>
<td>$L(G)$ finite</td>
<td>$L_1 \cap L_2 = \emptyset$ for non-ambiguous languages $L_1 \cdot L_2$</td>
</tr>
<tr>
<td>$D_1 = \Sigma^*$</td>
<td>$L_1 = \Sigma^*$ if $</td>
</tr>
<tr>
<td>$L_1 \subseteq R$</td>
<td>$L_1 = L_2$ if $</td>
</tr>
<tr>
<td></td>
<td>$L_1 \subseteq L_2$ if $</td>
</tr>
<tr>
<td></td>
<td>$L_1 = R$ if $</td>
</tr>
<tr>
<td></td>
<td>$R \subseteq L_1$ if $</td>
</tr>
</tbody>
</table>

where $L_1, L_2$ are context-free languages; $D_1, D_2$ are DCFL languages; $R$ is a regular language; $G$ is a context-free grammar, $w \in \Sigma^*$.
• Recall: Turing machines and Turing computability
• Register machines (LOOP, WHILE, GOTO)
• Recursive functions
• The Church-Turing Thesis
• Computability and (Un-)decidability
• Complexity
• Brief outlook: 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
- Brief outlook: other computation models, e.g. Büchi Automata