Advanced Topics in Theoretical Computer Science

Part 4: Computability and (Un-)Decidability (3)

17.01.2018

Viorica Sofronie-Stokkermans

Universität Koblenz-Landau

e-mail: sofronie@uni-koblenz.de

Last time

The Post Correspondence Problem

Post Correspondence Problem

Definition

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

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

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

A partial solution is an index set I such that

p_I is a prefix of q_I or q_I is an 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.

Post Correspondence Problem

Theorem. Assume $|\Sigma| \geq 2$.

(1) The Post Correspondence Problem with given start is undecidable.

(2) The Post Correspondence Problem is undecidable.

Post Correspondence Problem

Theorem. Assume $|\Sigma| \geq 2$.
(1) The Post Correspondence Problem with given start is undecidable.
(2) The Post Correspondence Problem is undecidable.

Today

Applications

Undecidabile problems in formal languages

Undecidabile problems in formal languages

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 $T = \{(u_1, v_1), \dots, (u_n, v_n)\}$ a CS over Σ_1; $\quad \Sigma' = \Sigma_1 \cup \{a_1, \dots, a_n\}$.

$L_{T,1} = \{a_{i_m} \dots a_{i_1} u_{i_1} \dots u_{i_m} | m \geq 1, 1 \leq i_j \leq n\}$ generated by c.f. grammar $G_{T,1}$.

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

$L_{T,2} = \{a_{i_m} \dots a_{i_1} v_{i_1} \dots v_{i_m} | m \geq 1, 1 \leq i_j \leq n\}$ generated by c.f. grammar $G_{T,2}$.

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

$G_{T,1}, G_{T,2}$ are unambigous.
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.

Undecidable problems in formal languages

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

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

T has a solution iff $\exists k{\geq}1\, \exists i_1, \dots, i_k$: $u_{i_1} \dots u_{i_k} = v_{i_1} \dots v_{i_k}$
iff $\exists k{\geq}1\, \exists i_1, \dots, i_k$: $u_{i_1} \dots u_{i_k} a_{i_k} \dots a_{i_1} = (a_{i_1} \dots a_{i_k} v_{i_1}^R \dots v_{i_k}^R)^R$
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!

Undecidable problems in formal languages

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 Σ. Then $L_1 \cap L_2 = \emptyset$ iff $\overline{L_1 \cap L_2} = \Sigma^*$ 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 Σ 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 4)

Undecidable problems for $\mathcal{L}_2$

decidable	**undecidable**	
$w \in L(G)$	G ambiguous	
$L(G) = \emptyset$	$D_1 \cap D_2 = \emptyset$	
$L(G)$ finite	$L_1 \cap L_2 = \emptyset$	for non-ambiguous languages $L_1 . L_2$
$D_1 = \Sigma^*$	$L_1 = \Sigma^*$	if $\|\Sigma\| \geq 2$
$L_1 \subseteq R$	$L_1 = L_2$	if $\|\Sigma\| \geq 2$
	$L_1 \subseteq L_2$	if $\|\Sigma\| \geq 2$
	$L_1 = R$	if $\|\Sigma\| \geq 2$
	$R \subseteq L_1$	if $\|\Sigma\| \geq 2$

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, $w \in \Sigma^*$.

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

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