Sapienza
Università di Roma
Dipartimento di Informatica

FOUNDATIONS OF COMPUTER SCIENCE
LECTURE 12: Decidability

Prof. Daniele Gorla

Decision problems on languages

- We saw that we can concentrate on decision problems
- Hence, any problem can be represented by a language

- The typical decision problems on languages are:

1. Membership («does w belong to L?»)
2. Emptyness («is L empty?»)
3. Equivalence («is $L_1 = L_2$?»)

- Actually, since languages are typically infinite, the above questions are formulated in terms of language describers (e.g., automata or grammars)

- These decision problems on langugеs correspond to abstract questions on problems:

1. Membership corresponds to «is p a solution of P?»
2. Emptyness corresponds to «does P admit any solution?»
3. Equivalence corresponds to «are P and P' different formulations of the same problem?»

Membership problem for DFAs

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: The language $A_{\mathrm{DFA}} = \{\langle B, w\rangle \mid B \text{ is a DFA that accepts } w\}$ is decidable.

Proof

We present a TM M that decides A_{DFA}.

- (When M receives its input, it first determines whether it properly represents a DFA B and a string w; if not, it rejects.)
- Then M carries out the simulation of B:
  - Initially, B's state is its starting state and the head is placed on the leftmost symbol of w
  - States are updated according to B's transition function δ and the head is moved always one character to the right.
  - When M finishes processing the last symbol of w, it accepts if B is in an accepting state and rejects otherwise.

Q.E.D.

Membership problem for NFAs, REs and RGs

SAPIENZA
Università di Roma
Dipartimento di Informatica

<u>Cor.:</u> The language $A_{NFA} = \{\langle N, w\rangle \mid N \text{ is a NFA that generates } w\}$ is decidable.

Proof

The TM that decides A_{NFA} behaves as follows:

On input $\langle N, w\rangle$, where N is a NFA and w is a string:

- Convert N into an equivalent DFA D (by using the procedure given in class 2)
- Run M (of the previous slide) on input $\langle D, w\rangle$
- If M accepts, accept; if M rejects, reject. <u>Q.E.D.</u>

<u>Cor.:</u> The language $A_{\mathrm{REX}} = \{\langle R, w\rangle \mid R \text{ is a regular expression that generates } w\}$ is decidable.

Proof

Like in the previous proof, but by using the algorithm for turning R into a NFA (seen in class 3).

<u>Q.E.D.</u>

<u>Cor.:</u> The language $A_{\mathrm{RG}} = \{\langle G, w\rangle \mid G \text{ is a regular grammar that generates } w\}$ is decidable.

Proof

Like in the previous proof, but by using the algorithm for turning G into a NFA (seen in class 4).

<u>Q.E.D.</u>

Emptyness problem for DFAs

SAPIENZA
Università di Roma
Dipartimento di Informatica

<u>Thm.:</u> The language $E_{\text{DFA}} = \{\langle A\rangle \mid A \text{ is a DFA and } L(A) = \emptyset\}$ is decidable.

Proof

A DFA accepts some string iff it reaches an accept state from the start state by traveling along its arrows.

→ this is a graph reachability problem, that we can solve by a visit of the graph underlying the DFA

→ idea: mark states that you pass though so that you don't cross them anymore

On input $\langle A\rangle$, where A is a DFA:

- Mark the start state of A
- Mark any state that has an incoming transition from any state that is already marked, until no new state gets marked
- If no accept state is marked, accept; otherwise, reject.

Q.E.D.

REMARK: the same idea can be used for NFAs, REs, and RGs.

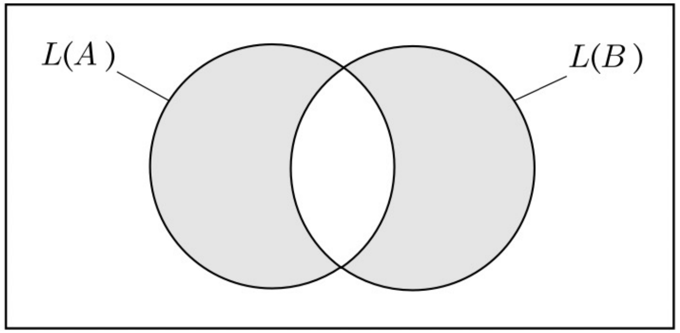
Equivalence problem for DFAs

Thm.: The language $EQ_{\text{DFA}} = \{ \langle A, B\rangle \mid A$ and B are DFAs and $L(A)=L(B)\}$ is decidable.

Proof:

Let us consider the language $C = (L(A)\setminus L(B)) \cup (L(B)\setminus L(A))$.

Pictorially, C is the grey part of this figure:



Now, $L(C) = \emptyset$ iff $L(A) = L(B)$; furthermore, C is regular

(see class 2 and the closure properties of regular lang's).

So, the TM for deciding EQ_{DFA}

- builds the NFA for C, starting from the DFAs A and B.

  (these constructions have been given in class 2, when presenting closure properties of RLs)
- it turns the NFA for C into a DFA
- behaves like the TM for deciding E_{DFA}.

Q.E.D.

Membership problem for CFLs (1)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

- One (bad) idea is to convert a PDA directly into a TM.

  → not hard to do because simulating a stack with the TM's more versatile tape is easy.
- The problem is that PDAs are nondeterministic and with ε-moves; hence, some branches of their computation may go on forever, reading and writing the stack without halting.

  → The simulating TM would have some non-halting branches (would not be a decider)
- Let's use CFGs for representing the CFL
- Another bad idea is to try all possible derivations in G

  → if $w \notin L(G)$ there is the risk of running forever

- Let us consider CFGs in ***Chomsky normal form***, that requires every rule to be of the form

  $$A ::= BC \quad \text{or} \quad A ::= a \text{, for } a \in \Sigma, A \in V, \text{ and } B,C \in V \setminus \{S\}$$

  In addition, we permit the rule $S ::= \varepsilon$ (where S is the start variable).
- Deriving a string long n (>0) in a CFG in Chomsky N.F. exactly requires $2n - 1$ steps ($S' \Rightarrow^{n-1} A_1 \ldots A_n \Rightarrow^{n} w_1 \ldots w_n$).
- Any CFG can be algorithmically turned into an equivalent one in Chomsky N.F.:
  - we add a new start variable S_0 and a new rule $S_0 ::= S$ (so that S_0 does not apper in any RHS)
  - we eliminate all ε-rules $A ::= \varepsilon$ by adding RHSs obtained by deleting every occurrence of A in every RHS
  - we eliminate all unit rules $A ::= B$ by adding to A all the RHSs that B produces
  - we convert the remaining rules to have a RHS with exactly 2 variables (by adding new variables)

Membership problem for CFLs (2)

EXAMPLE:

$$S \to ASA \mid aB \qquad A \to B \mid S \qquad B \to b \mid \varepsilon$$

→

$$S_0 \to S \qquad S \to ASA \mid aB \qquad A \to B \mid S \qquad B \to b \mid \varepsilon$$

→

$$S_0 \to S \qquad S \to ASA \mid aB \mid a \qquad A \to B \mid S \mid \varepsilon \qquad B \to b \mid \varepsilon$$

→

$$S_0 \to S \qquad S \to ASA \mid aB \mid a \mid SA \mid AS \mid S \qquad A \to B \mid S \mid \varepsilon \qquad B \to b$$

→

$$S_0 \to S \qquad S \to ASA \mid aB \mid a \mid SA \mid AS \mid S \qquad A \to B \mid S \qquad B \to b$$

→

$$S_0 \to S \mid ASA \mid aB \mid a \mid SA \mid AS \qquad S \to ASA \mid aB \mid a \mid SA \mid AS \qquad A \to B \mid S \qquad B \to b$$

→

$$S_0 \to ASA \mid aB \mid a \mid SA \mid AS \qquad S \to ASA \mid aB \mid a \mid SA \mid AS \qquad A \to B \mid S \mid b \qquad B \to b$$

→

$$S_0 \to ASA \mid aB \mid a \mid SA \mid AS \qquad S \to ASA \mid aB \mid a \mid SA \mid AS \qquad A \to S \mid b \mid ASA \mid aB \mid a \mid SA \mid AS \qquad B \to b$$

→

$$S_0 \to AA_1 \mid UB \mid a \mid SA \mid AS \qquad S \to AA_1 \mid UB \mid a \mid SA \mid AS \qquad A \to b \mid AA_1 \mid UB \mid a \mid SA \mid AS$$

$$A_1 \to SA \qquad U \to a \qquad B \to b$$

Membership problem for CFLs (3)

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: The language $A_{\mathrm{CFG}} = \{\langle G, w\rangle \mid G \text{ is a CFG that accepts } w\}$ is decidable.

Proof

On input $\langle G, w\rangle$, where G is a CFG and w is a string:

- Convert G into an equivalent grammar in Chomsky normal form
- If $|w| = 0$, then list all derivations with one step
  Otherwise list all derivations with $2|w| - 1$ steps
- If any of these derivations generate w, accept; if not, reject.

Q.E.D.

Emptyness problem for CFLs

SAPIENZA
UNIVERSITÀ DI ROMA
DIPARTIMENTO DI INFORMATICA

Thm.: The language $E_{\mathrm{CFG}} = \{\langle G\rangle \mid G \text{ is a CFG and } L(G) = \emptyset\}$ is decidable.

Proof

The language of a grammar is empty if and only if its start variable cannot generate a string of terminals.

The algorithm that we devise solves a more general problem: It determines *for each variable* whether that variable is capable of generating a string of terminals or not

On input $\langle G\rangle$, where G is a CFG:

- Mark all terminal symbols in G
- Mark any variable A for which G has a rule $A ::= U_1U_2\ldots U_k$ and each symbol $U_1, \ldots, U_k$ has already been marked, until no new variable gets marked
- If the start variable is not marked, accept; otherwise, reject.

Q.E.D.

REMARK: since there is an algorithm for turning a PDA into a CFG (see class 6), we also have that A_{PDA} and E_{PDA} are decidable.

Equivalence problem for CFLs

Let us now turn to $EQ_{\text{CFG}} = \{ \langle G, H \rangle \mid G$ and H are CFGs and $L(G) = L(H)\}$.

We could try to use the same technique for regular languages, i.e. by studying the emptyness of $(L(G)\backslash L(H)) \cup (L(H)\backslash L(G))$.

However, one crucial step in the proof for regular languages does NOT hold for CF ones.

Indeed, it is NOT true that «L and L' are CF implies that $L\backslash L'$ is CF».

To see this, consider $L = \{a^n b^n c^m \mid n,m \geq 0\}$ and $L' = \{a^m b^n c^n \mid n,m \geq 0\}$.
They are both CF, but $L\backslash L' = \{a^n b^n c^m \mid n,m \geq 0$ and $n \neq m\}$ is not CF.

Maybe, we can find another technique for proving EQ_{CFG} decidable

→ we shall see that this is NOT the case, and EQ_{CFG} is an example of *undecidable* language

→ equivalence of two CFGs (or PDAs) cannot be solved by ANY algorithm

Membership problem for CSLs

Thm.: The language $A_{LBA} = \{\langle N, w\rangle \mid N \text{ is a LBA that accepts } w\}$ is decidable.

Proof

The key observation is that N has exactly $|Q||w||\Gamma|^{|w|}$ distinct configurations (possible state, possible head position, possible content of the tape).

Then, the algorithm that decides A_{LBA} is the following:

On input $\langle N, w\rangle$, where N is an LBA and w is a string:

1. Simulate N on w for $|Q||w||\Gamma|^{|w|}$ steps or until it halts

2. If N has halted, accept if it has accepted and reject if it has rejected.

Otherwise, reject.

Indeed, if N on w has not halted within $|Q||w||\Gamma|^{|w|}$ steps, there must be a repeating configuration

→ because of determinism, N loops

Q.E.D.

We shall see in the next class that both the emptyness and the equivalence problem for CSLs is undecidable.