Decidable Languages

Introduction and Motivation

In this lecture we review some decidable languages related to regular and context free languages.

Later we will present an undecidable language.

Decidable Languages (rerun)

A language L over Σ^* is ***decidable*** if there exists a Turing machine recognizing L , that ***stops on every input***.

Objects as Input

The input for a TM is always a string. If we want to give some other object, e.g. an automaton or a grammar, the object must be encoded as a string. Encoding can be straight forward.

For example: An undirected graph G, is encoded by specifying its nodes and its edges. G's encoding is denoted as $\langle G \rangle$.

Decision Problems on Automata

Now we turn do deal with some problems related to regular and CF Languages. Typical problems are:

- Deciding whether a DFA accepts a language.
- Deciding whether a language is empty.
- Deciding whether two languages are equal, etc.

Finite Automata are Decidable

Consider the language

$$A_{DFA} = \{\langle B, w\rangle | B \text{ is a } DFA \text{ accepting } w\}$$

Note that we formulate a computational problem as a membership problem.

Theorem

A_{DFA} is a decidable language.

Proof

Consider

M="On a < B, w > input,

where B is DFA, w is string:

1. Simulate B on input w .

2. If B accepts - *accept*. Otherwise – *reject*."

Implementation

The encoding of $\langle B, w \rangle$ can be straight forward:

The five components of B are listed on M's tape one after the other.

TM M starts its computation by verifying that the encoding is well formed.

If this is not the case, the input is rejected.

Implementation

Following that, M simulates B's computation on w in a way, very similar to the way a computer program will do.

Simulation of a DFA Initialization

Assume that the DFA is encoded by a list of its components. TM M should first verify that the string representing B is well formed. Than it should use a "state dot" to mark B's initial state as its current state and an "input dot" to mark w's first symbol as the current input symbol.

Simulation of a DFA

Following that, M scans the substring representing B's transition function to find the transition that should take place for the current state and the current input symbol. Once the right transition is found, the new current state is known.

Simulation of a DFA

At this point, M moves the "state dot" from the previous current state, to the new current state, and moves the "input dot" from the previous input symbol to the next input symbol.

This procedure repeats until the input is finished. If at this stage B's current state is accepting, M accepts, otherwise it rejects.

Finite Automata are Decidable

Now we turn to consider the language

$$A_{NFA} = \{\langle B, w\rangle | B \text{ is a NFA accepting } w\}$$

and prove:

Theorem

A_{NFA} is a decidable language.

Proof

Consider

N="On a <NFA, string> input $\langle B, w \rangle$:

1. Convert NFA B to an equivalent DFA C.
2. Run TM M (See previous proof) on input $\langle C, w \rangle$.
3. If M accepts - *accept*. Otherwise – *reject*."

Implementation

Item 2 N's high level description says: "Run TM M on input $\langle C, w \rangle$."

Here M is used by N as a ***procedure***. This can be done as follows:

1. N is equipped with an additional tape on which M's input will be written.

Implementation

2. At the point in which M is called, a section in which M's input is written on the additional tape is added to N.
3. Following this section we add to N a complete copy of M, using its input tape.
4. This procedure is repeated on each call to M.

Regular Expressions are Decidable

An additional way to describe Regular Languages is by use of regular expressions.

Now we consider the language

$$A_{REX} = \{\langle R, w\rangle | R \text{ is an RE generating } w\}$$

and prove:

Theorem

A_{REX} is a decidable language.

Proof

Consider

P="On a <RE, string> input $\langle R, w \rangle$:

1. Convert RE R to an equivalent NFA A.
2. Run TM N (See previous proof) 0n w.
3. If N accepts - $accept$. Otherwise – $reject$."

The Emptiness Problem for DFA-s

In this problem it is required to compute whether a given DFA accepts at least one string: Consider the language

$$E_{DFA} = \{\langle A \rangle | A \; is \; a \; DFA \; and \; L(A) = \emptyset\}$$

and prove:

Theorem

E_{DFA} is a decidable language.

Proof

Consider

T="On a DFA input <A> :

1. Mark the start state of A.
2. Repeat until no new states are marked:
   3. Mark any state that has an incoming transition from an already marked state.
4. If no accept state is marked - *accept*. Otherwise – *reject*."

DFA Equivalence is Decidable

Our survey of decidability problems for regular languages is completed by considering

$$EQ_{DFA} = \{\langle A, B\rangle | A \text{ and } B \text{ are DFAs and } L(A) = L(B)\}$$

and proving:

Theorem

EQ_{DFA} is a decidable language.

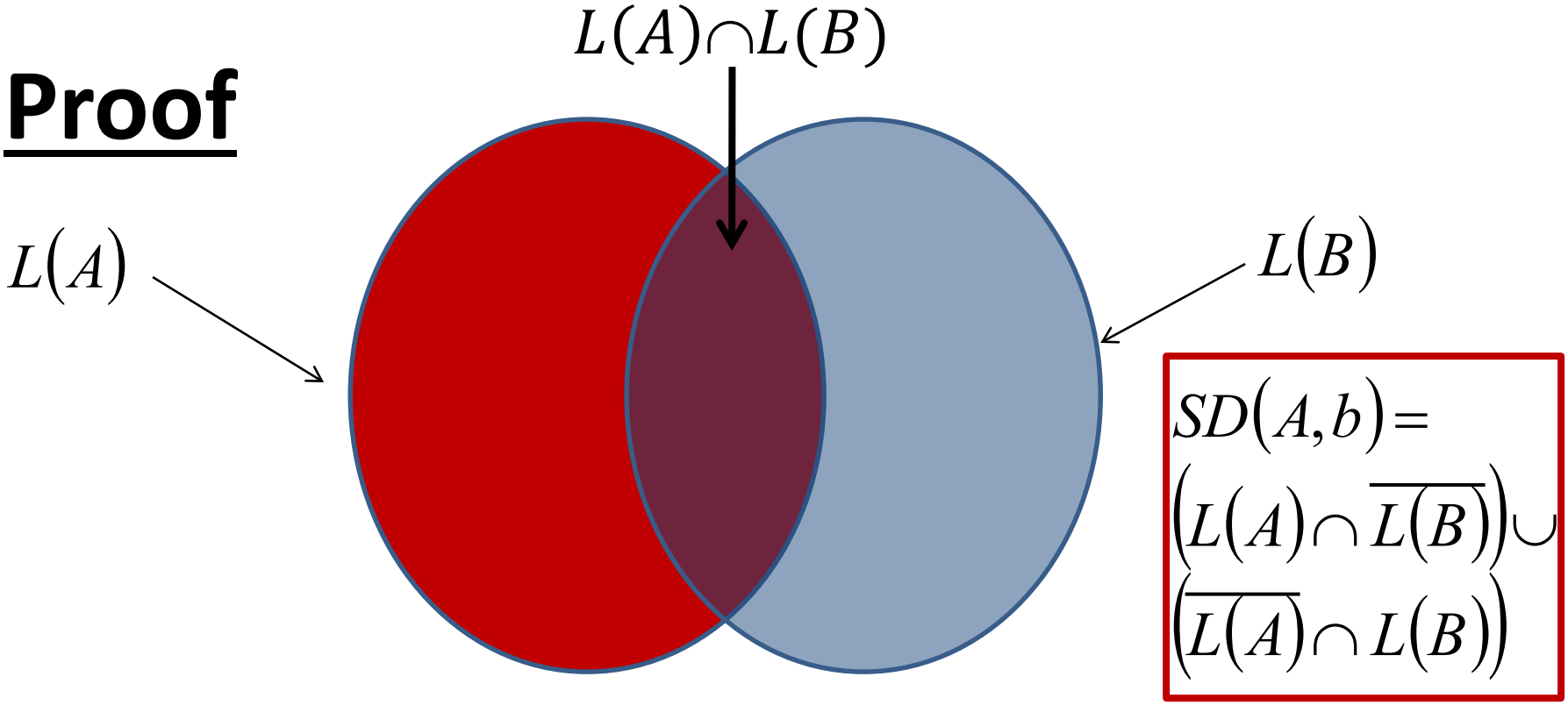
Proof

In order to prove this theorem we use the TM T of the previous proof. The input for T is a DFA C satisfying:

$$L(C) = \left(L(A) \cap \overline{L(B)}\right) \cup \left(\overline{L(A)} \cap L(B)\right)$$

This expression is called ***the symmetric difference of*** $L(A)$ and $L(B)$ and it can be proved that $L(C) = \phi$ iff $L(A) = L(B)$.

Proof



DFA C is constructed using the algorithms for constructing ***Union***, ***Intersection***, and ***Complementation***, Of Chapter 1.

Proof

The TM machine F for Deciding EQ_{DFA} gets as input to DFA-s A and B. This machine first activates the algorithms of Chapter 1 to construct DFA C and then it calls TM T to check whether $L(C) = \phi$. If T accepts so does F. Otherwise F rejects.

Decidable CFG Related Languages

Now we consider the language

$$A_{CFG} = \{\langle G, w\rangle | G \text{ is a CFG generating } w\}$$

and prove:

Theorem

A_{CFG} is a decidable language.

Decidable CFG Related Languages

If we try to prove this theorem by checking all possible derivations of <G>, we may run into trouble on grammars containing ***cycles***, e.g.

$$A \to B$$
$$B \to A$$

This grammar has an infinite derivation which is hard to deal with.

Decidable CFG Related Languages

The way to solve this problem is by using the following definition and theorem:

Definition

A context-free grammar is in ***Chomsky normal form*** (CNF) if every rule is of the form:

$$A \to BC$$

$$A \to a$$

where A, B, C are variables (B and C may not be the start variable) and a is a terminal. Also, the rule $S \to \varepsilon$ is permitted if S is the start variable.

Why Chomsky Normal Form?

The key advantage is that in Chomsky normal form, every derivation of a string of n letters has exactly 2n − 1 steps.

Thus: one can determine if a string is in the language by exhaustive search of all derivations.

Decidable CFG Related Languages

Theorem

Every context-free grammar has an equivalent grammar in Chomsky normal form.

$$A \to BC$$

$$A \to a$$

Note: In a CNF grammar every non-final derivation **extends** the current string.

Decidable CFG Related Languages

Result

If G is a context-free grammar in CNF then:

1. A derivation of 0-length string w takes a single production.
2. A derivation of an n-length string w takes a $2n$-1 productions.

Proof of Theorem

Consider

S="On <CFG, string> input $\langle G, w \rangle$:

1. Convert G to an equivalent CNF grammar.
2. List al all derivations with $2n$-1 productions.
3. If w is generated by one of these derivations - *accept*. Otherwise – *reject*."

The Emptiness Problem for CFL-s

Consider the language

$$E_{CFG} = \{\langle G \rangle | G \text{ is a CFG and } L(G) = \emptyset\}$$

Theorem

E_{CFG} is a decidable language.

How can this be proved?

The Emptiness Problem for CFL-s

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

1. Mark all terminal symbols in G.
2. Repeat until no new variables are marked:
   3. Mark any variable A where G has a rule of the form $A \to U_1 U_2 \cdots U_k$ and $U_1, U_2, \cdots, U_k$ are all marked.
4. If the start variable is not marked *accept*

   otherwise *reject*."

CFG Equivalence is not Decidable

Consider the language

$EQ_{CFG} = \{\langle G, H\rangle | G \text{ and } H \text{ are CFGs and } L(G)=L(H)\}$

Unlike DFA-s CFL-s are not closed under intersection and complementation. Therefore we cannot decide EQ_{CFG} using the method we used for EQ_{DFA} . In the near future we will prove:

Theorem

EQ_{CFG} is an undecidable language.

CFL-s are Decidable

Our survey of decidability problems for

CFL-s/CFG-s is completed by considering the decision problem for context Free Languages, namely: Given a context free language L, does there exist a TM that decides L?

CFL-s are Decidable

Recall that every CFL is recognized by some PDA.

It is not hard to prove that a TM can simulate a PDA, but this is not enough.

For many CFL-s the PDA recognizing them is nondeterministic, which may cause the following problem:

CFL-s are Decidable

Let L be a CFL, let p a be a PDA recognizing L and let w be a string such that $w \notin L$.

PDA P may never stop on w and a TM simulating P may loop while a decider should stop on every input. Nevertheless there is a solution:

Theorem

Let L be a CFL. There exists a TM deciding L.

Proof

Since L is a CFL, there exist a CFG G that generates L. The problem is solved by TM M_G that contains a copy of G, as follows.

$M_G =$ "On input w:

1. RUN TM S on input $\langle G, w \rangle$
2. If S accepts - *accept*. Otherwise – *reject*."

The relationship among classes of languages