

Lecture Notes for CS 509

Foundations of Computer Science

Lecture 1

Lecturer: Joe Kilian
Scribe: Fengming Wang
Department of Computer Science
Rutgers, The State University of New Jersey

1 What is a *Set*?

Definition (From Webster). A *set* is a collection of elements and especially *mathematical* ones (as numbers or points), – called also class.

The previous definition seems to be "scary" and not exact, let us check out a more formal one as follows.

Definition (From <http://mathworld.wolfram.com/Set.html>). A *set* is a finite or infinite collection of objects in which order has no significance, and multiplicity is generally also ignored (unlike a list or multiset). Members of a set are often referred to as *elements* and the notation $a \in A$ is used to denote that a is an element of a set A .

For example, Integers, $\{1, 3, 2\}$, $\{ab, abc\}$ and $\{\{1\}, \{1, 2\}, \{1, 3\}\}$ are all sets.

2 What is a *Language*?

Definition (From Joe). A *language* is a subset, possibly infinite, of finite length strings over some alphabet Σ .

For example, $L = \{a, aa, aaa, \dots\}$.

3 Diagonalization argument

3.1 Diagonalization on sets

Now suppose we have the following sets:

- $A = \text{set of all finite sets,}$
- $B = \text{set of all infinite sets,}$
- $C = \text{set of all sets that do not contain themselves.}$

Problems arised:

<i>Question</i>	<i>Answer</i>
$A \in A ?$	<i>no</i>
$B \in B ?$	<i>yes</i>
$A \in C ?$	<i>yes</i>
$B \in C ?$	<i>no</i>
$C \in C ?$	<i>...</i>

So be watchful about this kind of holes(Known as Russell's Antinomy, check out <http://mathworld.wolfram.com/RussellsAntinomy.html>), but on the other hand, it could be very helpful if we try to demonstrate that some set is beyond the scope of countability.

3.2 Diagonalization on languages

According to the standard lexicographic order, we may enumerate finite strings of all lengths. But what about languages? Can we somehow list all of the languages?

The answer is no and we prove it via contradiction.

First of all, the standard lexicographic order gives us a bijection between set of finite strings and set of integers. Therefore each language is thought of as a set of natural numbers.

Second, suppose all of languages could be listed in some way. With respect to this particular enumeration, we create the following infinite N by N table. For each entry (i, j) , we indicate 1 if $j \in L_i$, 0 otherwise.

	0	1	2	3	...
L_0	1	0	1	0	...
L_1	0	1	1	1	...
L_2	1	1	1	1	...
L_3	0	1	1	0	...
...

Define an "evil" language L_{evil} as follows: $L_{evil} = \{i \mid i \notin L_i\}$

Question: can you specify the index of L_{evil} within this enumeration?

4 Regular languages

Regular languages could be defined by regular expressions as well as finite automata. For details, please refer to the textbook.

4.1 Operations

1. Union

Given $L_1 = \{a, b\}$, $L_2 = \{c, \lambda\}$, $L_1 \cup L_2 = \{a, b, c, \lambda\}$.

Regular languages are closed under union.

2. Concatenation

Given $L_1 = \{a, b\}$, $L_2 = \{c, \lambda\}$, $L_1L_2 = \{a, b, ab, ac\}$.

Regular languages are closed under concatenation.

3. Kleene star

Given $L = \{a\}$, $L^* = \{\lambda, a, aa, aaa, \dots\}$.

Regular languages are closed under Kleene star operation.

4. Complementation

Regular languages are closed under complementation.

5. Intersection

Due to 1, 4, and the fact that $L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$, regular languages are closed under intersection.

These closure properties are proved via finite automata. For details, see textbook.

4.2 Proving membership via regular expression

Basically we break down the given string and derive a series of steps which demonstrates that every part is consistent with each corresponding operation of the regular expression. For example, Suppose we have a given string $aaabb$ and a given language described by a^*b^* .

$$\begin{array}{ccc}
a \in L_a = \{a\} & & b \in L_b = \{b\} \\
\downarrow & & \downarrow \\
aaa \in L_a^* & & bb \in L_b^* \\
\downarrow & & \downarrow \\
& \downarrow & \\
& aaabb \in L_a^* L_b^* &
\end{array}$$

Question: how do you disprove membership?

5 Finite automata

Finite automata are alive! They

<i>have eyes</i>	→	<i>see the current symbol</i>
<i>have legs</i>	→	<i>walk along the designated transition</i>
<i>have a little bit of brain</i>	→	<i>choose the next state</i>
<i>can smile</i>	→	<i>when reaching final state at the end</i>
<i>can frown</i>	→	<i>otherwise</i>

There are several variations: Deterministic Finite Automata(*DFA*), Nondeterministic Finite Automata(*NFA*), Probabilistic Finite Automata(*PFA*), etc.

From the textbook, we know that *DFA* and *NFA* are equivalent via a not so efficient transformation, namely, exponentiation of states. Question: is exponentiation necessary?

6 Probabilistic finite automata

6.1 Definition

Given a *PFA* M and input x , let

$$M(x) = \text{Probability that } M \text{ ends up in an accept state on input } x$$

Define L by (M, λ) such that $x \in L$ iff $M(x) \geq \lambda$, where the real number λ is usually called *cut point*.

Based on the definition, it is easily seen that there are uncountably many languages recognized by *PFA*. Let us take a look at the following example.

Suppose $\Sigma = \{0, 1\}$ and x be any arbitrary finite string, there exists M such that $M(x) = 0.x$ (The construction of M is very simple). Moreover, for any distinct λ , $L_{M,\lambda}$ is distinct. The reason is that for any two distinct $\lambda_1 < \lambda_2$, we could always come up

with a binary number x' such that $\lambda_1 < x < \lambda_2$. By the property of M , we know that $x \in L_{M,\lambda_1}$ but $x \notin L_{M,\lambda_2}$.

6.2 PFA with gap

Gap property means that there exists a constant ϵ , for every input x such that either $M(x) \geq \lambda + \epsilon$ or $M(x) \leq \lambda - \epsilon$. We remark that as long as ϵ is a constant, even if it is very small, we could always amplify it close to $\min\{\lambda, 1 - \lambda\}$ (One technique is to use majority vote and analyze it with Chernoff bound. For details, see <http://en.wikipedia.org/wiki/Chernoff-bound>).

Theorem 6.1. *The languages accepted by PFA with gap property are regular.*

Sketch of proof. Given L , suppose that the PFA for L has m states. Consider the probability distributions on these states. Denote each such distribution x as a real vector of dimension m , namely $(x_1, x_2, \dots, x_m) : \sum_{i=1}^m x_i = 1$.

On the other hand, choose a small constant γ such that $\gamma \ll \frac{\epsilon}{m}$. Define the following equivalence relation:

$x_1 \approx x_2$ if $\|x_1\| - \|x_2\| \leq \gamma$, where $\|x_1\|$ means the L_1 norm of x_1 .

Therefore we get a partition on all the probability distributions (There might be some boundary issues, but for simplicity, we just put the boundary into an arbitrary adjacent equivalence class). The number of equivalence classes are finite, depending on γ .

Given a finite string y , we perform a walk starting from distribution x according to PFA and y , then we get another distribution on all the states and denote this distribution as (x, y) . We obtain the following claim.

Claim 6.1.1. *If $x_1 \approx x_2$, then for all finite strings y , $(x_1, y) \approx (x_2, y)$, which means that starting from either x_1 or x_2 , the PFA will accept the same language.*

Proof of claim.

$$\begin{aligned}
& \left| |(x_1, y)| - |(x_2, y)| \right| \\
&= \left| \sum_{i=1}^m \left| Pr[\text{in state } i \text{ after seeing } x_1, y] - Pr[\text{in state } i \text{ after seeing } x_2, y] \right| \right| \\
&= \left| \sum_{i=1}^m \left| \sum_{j=1}^m Pr[\text{in state } j \text{ after } x_1] Pr[j \text{ to } i \text{ from seeing } y] - \right. \right. \\
&\quad \left. \left. \sum_{j=1}^m Pr[\text{in state } j \text{ after } x_2] Pr[j \text{ to } i \text{ from seeing } y] \right| \right| \\
&\leq \sum_{i=1}^m \gamma \\
&< \epsilon
\end{aligned}$$

□

Thus we make the equivalence classes as the states of new DFA and design the corresponding transitions according to original PFA. □