SAN JOSE STATE UNIVERSITY
DEPARTMENT OF ELECTRICAL ENGINEERING

CS 154 Formal Languages and Computability Spring 2012
Section 1 Room MH 225 Class 2: 01-30-12

T. Howell

<u>Summary of Class 1</u>

Decision problem is a function whose outputs are "yes" or "no".
We need to know
    the set A of all possible inputs.
    the set $B \subseteq A$ of "yes" instances.

Definitions given:
An <u>alphabet</u> is any finite set of characters.
A <u>string</u> over Σ is any finite length sequence of elements of Σ.
The <u>length</u> of a string x is denoted $|x|$.
The string of length 0 is called the <u>null string</u> and is denoted ε. (not $\in$). Thus $|\varepsilon| = 0$.
We write a^n for a string of n a's. Example a^4 = aaaa.
The set of all strings over alphabet Σ is called Σ^*.

Operations on strings.

Concatenation
We write x^n for the concatenation of n copies of the string x.
Notation: If $a \in \Sigma$ and $x \in \Sigma^*$, we write #a(x) for the number of a's in x.
A <u>prefix</u> of a string x is an initial substring.
A <u>proper prefix</u> of x is one other than x or ε.

<u>New material</u>

Differences between strings and sets:
Strings have order, and repetition matters. Sets are unordered, and repetition doesn't matter. $\{a, b\} = \{b, a\}$, but ab $\neq$ ba. $\{a, a, b\} = \{a, b\}$ but aab $\neq$ ab.

Operations on sets.
$|A|$ is cardinality (note overlapping notation with length of a string.)
Union
Intersection
Complement
Set concatenation

Powers (by concatenation)
Asterate (*)

Many algebraic properties are enumerated in the book.
De Morgan laws:
$\sim(A \cup B) = \sim A \cap \sim B$
$\sim(A \cap B) = \sim A \cup \sim B$

Finite Automata and Regular Sets

The <u>state</u> of a system is an instantaneous description containing all information necessary to determine its future behavior.
A <u>transition</u> is a change of state.
Our abstract machines make instantaneous transitions. Real world machines with states and transitions: digital circuits, watches, elevators, games (chess, solitaire, ...),
If there are finitely many states and transitions, we call it a <u>finite state transition system</u>.
Our abstract model is called a <u>finite automaton</u>.

Formal definition:
$M = (Q, \Sigma, \delta, s, F)$ (a 5-tuple)

Q = set of states
Σ = input alphabet
δ = transition function $\delta: Q \times \Sigma \to Q$ (explain notation)
s = start state $s \in Q$
F = accept states = final states $F \subseteq Q$

This is the formal description used to prove things.

Example 1:
$Q = \{0, 1, 2, 3\}$
$\Sigma = \{a,b\}$
$s = 0$
$F = \{3\}$
$\delta(0,a) = 1$
$\delta(1,a) = 2$
$\delta(2,a) = \delta(3,a) = 3$
$\delta(q,b) = q$ for all $q \in Q$

Other ways to write down a finite automaton:

Table:

		a	b
→	0	1	0
	1	2	1
	2	3	2
	*3	3	3

Transition diagram:

Accepting (or final) states are marked * in the table and circled in the transition diagram.

Explain how the FA operates. Put a pebble on state s. Move it around according to δ as you scan the input string one symbol at a time. When the end is reached, see if the pebble is on a state belonging to F. If so it is <u>accepted</u>, otherwise it is <u>rejected</u>.

Example: baabbaab is accepted.
babbbab is rejected

We can see that any string with 3 or more a's will be accepted.

Back to formal methods again:
Define $\hat{\delta}: Q \times \Sigma^* \to Q$

$\hat{\delta}(q, \varepsilon) = q$

$\hat{\delta}(q, xa) = \delta(\hat{\delta}(q, x), a)$

This is another inductive definition. Work through it carefully.
Note that $\hat{\delta}$ works with strings of any length, any element of Σ^*, while δ just works with individual symbols (= strings of length 1). But they agree on strings of length 1.

A string is accepted by automaton M if $\hat{\delta}(s, x) \in F$. Otherwise it is rejected.

The <u>language</u> accepted by M is the set of strings accepted by M and is denoted L(M).
$L(M) = \{x \in \Sigma^* \mid \hat{\delta}(s, x) \in F\}$ (quick aside to explain set builder notation)

A subset A $\subseteq \Sigma^*$ is <u>regular</u> if A = L(M) for some finite automaton M.

For our example, the automaton accepts
{x ∈ {a,b}* | x contains at least 3 a's},
so this is a regular set.

Example 2
Consider the set

RLL(1, 3) = $\{x \in \{0,1\}^* \mid 1x$ has 1, 2, or 3 0's between adjacent 1's$\}$
= $\{x \in \{0,1\}^* \mid 1x$ does not contain 11 or 0000$\}$

Here is an automaton for this set.

		0	1
→	*0	1	4
	*1	2	0
	*2	3	0
	*3	4	0
	4	4	4

Draw transition diagram.
The automaton is in states 0, 1, 2, 3 when it has seen that many zeroes since the last 1 (or the beginning of the input). It is in state 4 when it has seen either an initial 1 or four consecutive 0's.
A subset of this language was used to encode data in disk drives (1970's).