Lecture 7

Scribe Notes for ECS 20 (Prof. Rogaway)
Scribe: Charles Hollister

Today:

1. Back to Sets
2. Languages
3. Regular Languages

Basic Set Operation:

Recall the basic set operator, $\in$. From this operator come other set quantifiers and operations:

$\subseteq , \supseteq$

$\subsetneq , \supsetneq$

$\cup , \cap$

$\backslash$ "Set difference" (sometimes denoted – , a minus sign)

$\oplus$ - symmetric difference (xor for sets, denoted Δ in your book and sometimes denoted (circled v))

x – Cartesian product – A x B is the set of ordered pairs (a,b) with the 1st element in the first set and the 2nd element in the 2nd set

P – Power set operator, unary operator (takes 1 input). ***P***(x) is the "set of all subsets of x"

P(X) = {A:A$\subseteq$X}

More on power sets:

For example, take X={0,1,2}, then

P(x)={ϕ,{0},{1},{2},{0,1},{0,2},{1,2},{0,1,2}}

We can more systematically enumerate the elements of this set by counting in binary just as in a truth table, indicating if the given element is (1) or is not (0) in the given set:

“2”	“1”	“0”	
0	0	0	ϕ
0	0	1	{0}
0	1	0	{1}
0	1	1	{0,1}
1	0	1	{0,2}
1	1	0	{1,2}
1	1	1	{0,1,2}

(Aside - Prof. Rogaway mentioned in class that we should structure our truth tables by counting in binary in the usual order, like above, as good practice for readability and to be systematic)

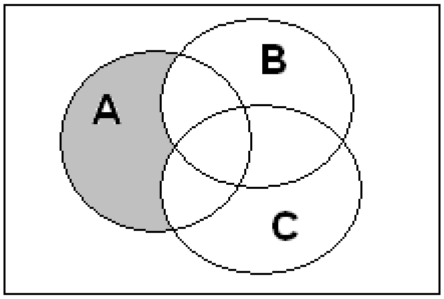
Notice in the table that the power set of X had 8 values in it. In general, if X is finite, then $|\boldsymbol{P}(X)| = 2^{|X|}$. Indeed sometimes $\boldsymbol{P}(X)$ is written 2^X as a reminder of how big this set is.

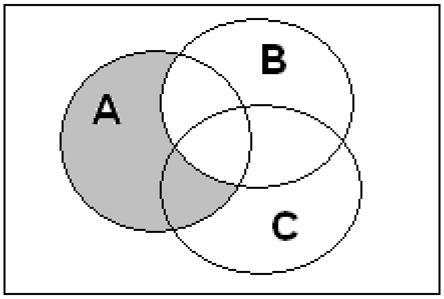
As another example, we talked abput $\boldsymbol{P}(N)$, where N is the set of natural numbers.

Using the other set operators:

Next, to help improve or facility with the operators on sets, we asked if the following claim is true or false:

$$(A \setminus B) \setminus C \overset{?}{=} A \setminus (B \setminus C)$$





We can see from the diagram that the statement is *false*. Instead, based on inspection, it would appear that:

$$(A \setminus B) \setminus C = A \setminus (B \cup C)$$

Proof: It is common to show set-equalities by arguing both inclulusions. But here we will do it all at once:

$x \in (A \backslash B) \backslash C$ iff $x \in A \backslash B \wedge x \notin C$ iff
$(x \in A) \wedge (x \notin B) \wedge (x \notin C)$ iff $x \in A \wedge \neg x \in B \wedge \neg x \in C$ iff
$x \in A \wedge \neg (x \in B \vee x \in C)$ iff $x \in A \wedge \neg (x \in B \cup C)$ iff
$\mathrm{x} \in A \backslash (B \cup C)$

Set vocabulary definitions

A few more words we often use:

- <u>singleton set</u> – a set with only one element.
- the <u>empty set</u> is the set with no elements. Denoted ∅. Note this symbol is not the same as a Greek letter phi. (Indeed I believe the origin is Norwegian).
- We define |A| as the number of elements in A (or <u>cardinality</u> of A), also written n(A) and #A
-Sets A and B are <u>disjoint</u> if A and B share no elements, ie, if their intersection is empty..
Classmate question – Does the "if" in the definition imply iff?
Answer – yes, definitions **always** "iff" when "if" is used.

Some formal language theory

Σ - an <u>alphabet</u> – an alphabet is a finite, nonempty set. Eg, {0, 1}, {a, b}, {0,1,2,3,4,5,6,7,8,9}, {1}, ASCII – these are all alphabets. Elements in an alphabet are called *characters* (or sometimes *symbols*).

- Is ∅ an alphabet? NO, it is empty. An alphabet must be nonempty.

- is **N**, the set of natural numbers, an alphabet? No, it is infinite. An alphabet must be finite.

A <u>string</u> is a finite sequence of characters. The characters all come from some understood alphabet.

Examples and more definitions

Σ^* represents **all** the strings over the alphabet Σ. This is an infinite set. But all the elements in this infinite set are strings of finite length. If

$$\Sigma = \{0,1\}$$

Then

$$\Sigma^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000, \ldots, 111, 0000, \cdots\}$$

In this case, ε represents the "empty string". It is the unique string of length 0. In the case of the English alphabet:

$$\Sigma = \{a, \ldots, z\}$$

Σ^* contains dog, fish, ε, *etc*

Suppose *x* and *y* are strings, $x, y \in \Sigma^*$,

- Then $x \circ y = xy$ denotes the characters of *x*, in order, followed by the characters of *y*, again in order This is known as the <u>concatenation</u> operator; we have <u>concatenated</u> the two strings.

Consider *x* = dog, *y* = fish, then *xy* = dogfish.

- The symbols | … |, when applied to strings, as in $|x|$, denotes the *length* of the string *x*. So the symbol has a different meaning when applied to sets and strings.

- True or False? A string exists of length ∞. <u>False</u>, strings are finite.

- True or False? There exists a unique string of length 1. The answer depends upon the set. For $\Sigma = \{1\}$, the unary alphabet, it's true. For alphabets with 2 or more characters, it is false.

- The exponential operator in strings is defined as in $1^3 = 111$, or more generally, $x^n = x\, x^{n-1}$ for n>0 and $x^0 = \varepsilon$.

- The reason we define the 0th power as the empty string, is so the last statement holds true for *n*=1, i.e. $x^1 = x \circ \varepsilon = x$.

- For the length operator, the following holds: $|xy| = |x| + |y|$. Clear?

- *x*[*i*:*j*] is often used to represent *substring* of *x* starting at position *I* and ending at position *j*, where $i \geq 1, j \leq |x|$.

For example, *x*=101101, $|x| = 6$; $x[2:5] = 0110$

X^R is used to represent the "reversal" of X.

$$(\text{cat})^R = \text{tac}, (\text{hello})^R = \text{olleh}$$

If $x = x^R$ the we call x a *palindrome*.

True or False – Palindromes are all of even length.
False, consider "1".

0 is both a character and a string: $0 \in \Sigma$
ε is only a string it is not a character.
01 is likewise a string, it is not a character.

Language – A set of strings, all over the same alphabet.

$\Sigma = \{0,1\}$

This is a language. It is also an alphabet.
All of the following are examples of languages:

$L = \{\varepsilon, 00, 01, 10, 11, 0000, 0001 \cdots\} = \{x \in \Sigma^* : |x| \text{ is even}\}$

$L = \{1^P : \text{p is prime}\} = \{11, 111, 11111, 1111111, \cdots\}$

$L = \{x \in \{0,1\}^* :$ x represents a prime number, no leading zeroes, written in binary$\}$

$= \{10, 11, 101, 111, 1011\}$

L={dog, cat, fish}

Languages can be finite or infinite.

Languages are sets, so set operators apply to languages. For example:

$L_1 = \{\text{dog,cat,fish}\}$

$L_2 = \{\text{dog,frog}\}$

$\left|\text{L}_1 \cup L_2\right| = 4$

$\left|\text{L}_1 \setminus L_2\right| = 2$

$\left|\text{L}_1 \oplus L_2\right| = 3$

The concatenate function can be lifted to apply to language; when L_1 and L_2 are languages, L_1 L_2 is defined as all the combinations of a string from L_1followed by a string from L_2. That is,

$$L_1L_2 = L_1 \circ L_2 = \{xy \mid x \in L_1, y \in L_2\}$$

With the set we have written above,

$L_1L_2 = \{\text{dogdog,dogfrog,catdog,catfrog,fishdog,fishfrog}\}$

True or False: $\varnothing$ is a language.

True – all its elements are strings (that is, all 0 of them).

$L\varnothing = \varnothing L = \varnothing$

$\{ \varepsilon \}$ is also a language – the singleton language containing just the empty string. Clearly

$$L\{\varepsilon\} = L$$

as

$$x\varepsilon = \varepsilon x = x$$

Student question:: is ε in every language?

Answer, No, we can choose to have it in a given language or not.

$L = \{1^i : i \text{ is even}\}=\{\varepsilon,11,1111,111111\}$

$x^0 = \varepsilon$

We can write L^2 for L L, and L^3 for L L L, etc. For the example just given, when L is all the even length strings of 1's, what is L^2 and L^3 . Just L.

True or False: if L contains the empty string, the *LL'* contains all of *L'*. True.

Taking it a step further, we can represent all the strings formed by concatenating string from the language as $L^* = \bigcup_{i \geq 0} L^i$

In order for that definition to make sense, we need to define L^0. $L^0 = \{\varepsilon\}$ is the only way to sensibly do this. If, for example we had set this to the emptyset, then L^* would always be empty, which would be bad.

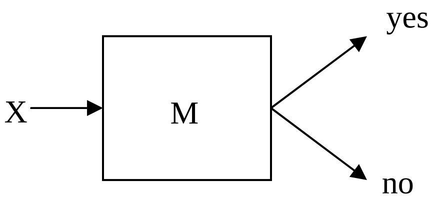
Say L={dog,cat}. Then L^* ={ ε, cat, dog, catcat, catdog, dogcat, dogdog, …}

$$\{0,1\}^* = \{\varepsilon, 0, 1, 00, 01, 10, 11, 000 \cdots\}$$

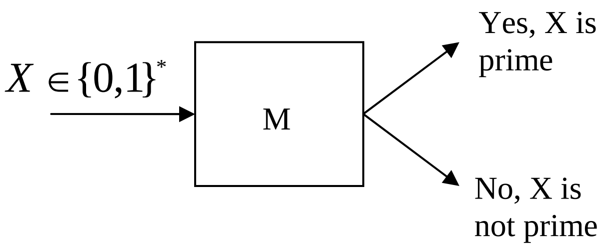
Just what we said earlier, but now seen from a more general light.

Why are computer scientists interested in languages?

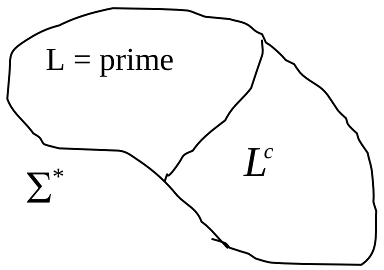
We can imagine a model for a computer program with an input of a string *X*. We define the language associated to the program by saying that $x \in L \Leftrightarrow M(X)$ says "yes"



That is to say, the computer program could decide if an input was part of the language. In this way, Languages correspond to decision questions. For example, a computer could take the input of a language X, and decide if the string represented by *X* is prime.



We can illustrate this as follows:



The program is splitting the universe into two sets: those it says “yes” to (they are in the language) and those that it says “no” to (they are not in the languge).

When the language L is finite, it is conceptually easy to write a program to decide it: just look it up in a table.

Other languages seem easy, too. For example,

$L_1 = \{1^i : i \text{ is even}\}$ is an “easy” language to make a machine to decide. One sense in which it is easy is that there’s a short piece of notation that a machine could interpret that would describe that language. The short piece of notation I have in mind is this: $(11)^*$. This notation is supposed to mean “the string 11, repeated any number of times”.

$L_2 = \{1^p : p \text{ is prime}\}$ is a much harder language to describe. It doesn’t seem like we could describe it by a short string using symbols like concatenation and union and this star operator.

Regular Languages

We will consider the kinds of languages that can be described using the following special symbols:

$( )\ \bigcup\ *\ \circ$

We also allow symbols from some underlying alphabet, and the empty string symbol. What we have just described is the vocabulary we will use for regular expression of languages. The “meaningful” strings over these symbols denote language according to natural rules:

$$L((\alpha \cup \beta)) = L(\alpha) \cup L(\beta)$$
$$L((\alpha \circ \beta)) = L(\alpha)L(\beta)$$
$$L((\alpha^*)) = (L(\alpha))^*$$

Also, the language associated to a symbol from the alphabet denotes that singleton language, and the empty string symbol denotes that singleton language.

We can use this more specifically to describe some languages with nice compact expressions. In our earlier example, (11)* denotes the lanaguge:

$(\{1\} \circ \{1\})^* = \{11\}^* = L_1$

Here’s another example:

$$0(0 \cup 1)^* 0 = \{x \in \{0,1\}^* : \text{ x starts and ends with a "0" and } |\text{x}| \geq 2\}$$

If we didn’t write the part about the length of x then the right hand side would include the string 0 but the lefth-hand side would not.

What if we wanted to make a regular expression for a binary string that started and ended with the same character, or with no character at all? That would be

$$\varepsilon \cup 1 \cup 1(0 \cup 1)^* 1 \cup 0 \cup 0(0 \cup 1)^* 0$$

Isn’t this fun?