Lecture 7

Pattern Matching

What happens when one types `rm *` in UNIX? (If you don't know, don't try it to find out!) What if the current directory contains the files

```
a.tex  bc.tex  a.dvi  bc.dvi
```

and one types `rm *.dvi`? What would happen if there were a file named `.dvi`?

What is going on here is *pattern matching.* The * in UNIX is a pattern that matches any string of symbols, including the null string.

Pattern matching is an important application of finite automata. The UNIX commands `grep`, `fgrep`, and `egrep` are basic pattern-matching utilities that use finite automata in their implementation.

Let Σ be a finite alphabet. A *pattern* is a string of symbols of a certain form representing a (possibly infinite) set of strings in Σ^*. The set of patterns is defined formally by induction below. They are either *atomic patterns* or *compound patterns* built up inductively from atomic patterns using certain *operators.* We'll denote patterns by Greek letters α, β, γ, ...

As we define patterns, we will tell which strings $x \in \Sigma^*$ *match* them. The set of strings in Σ^* matching a given pattern α will be denoted $L(\alpha)$. Thus

$$L(\alpha) = \{x \in \Sigma^* \mid x \text{ matches } \alpha\}.$$

In the following, forget the UNIX definition of *. We will use the symbol * for something else.

The *atomic patterns* are

- a for each $a \in \Sigma$, matched by the symbol a only; in symbols, $L(a) = \{a\}$;
- ϵ, matched only by ϵ, the null string; in symbols, $L(\epsilon) = \{\epsilon\}$;
- $\emptyset$, matched by nothing; in symbols, $L(\emptyset) = \varnothing$, the empty set;
- #, matched by any symbol in Σ; in symbols, $L(\#) = \Sigma$;
- @, matched by any string in Σ^*; in symbols, $L(@) = \Sigma^*$.

Compound patterns are formed inductively using binary operators $+$, $\cap$, and $\cdot$ (usually not written) and unary operators $^+$, *, and $\sim$. If α and β are patterns, then so are $\alpha + \beta$, $\alpha \cap \beta$, α^*, α^+, $\sim\alpha$, and $\alpha\beta$. The last of these is short for $\alpha \cdot \beta$.

We also define inductively which strings match each pattern. We have already said which strings match the atomic patterns. This is the basis of the inductive definition. Now suppose we have already defined the sets of strings $L(\alpha)$ and $L(\beta)$ matching α and β, respectively. Then we'll say that

- x matches $\alpha + \beta$ if x matches either α or β:

$$L(\alpha + \beta) = L(\alpha) \cup L(\beta);$$

- x matches $\alpha \cap \beta$ if x matches both α and β:

$$L(\alpha \cap \beta) = L(\alpha) \cap L(\beta);$$

- x matches $\alpha\beta$ if x can be broken down as $x = yz$ such that y matches α and z matches β:

$$\begin{aligned} L(\alpha\beta) &= L(\alpha)L(\beta) \\ &= \{yz \mid y \in L(\alpha) \text{ and } z \in L(\beta)\}; \end{aligned}$$

- x matches $\sim\alpha$ if x does not match α:

$$\begin{aligned} L(\sim\alpha) &= \sim L(\alpha) \\ &= \Sigma^* - L(\alpha); \end{aligned}$$

- x matches α^* if x can be expressed as a concatenation of zero or more strings, all of which match α:

$$\begin{aligned} L(\alpha^*) &= \{x_1 x_2 \cdots x_n \mid n \geq 0 \text{ and } x_i \in L(\alpha),\ 1 \leq i \leq n\} \\ &= L(\alpha)^0 \cup L(\alpha)^1 \cup L(\alpha)^2 \cup \cdots \end{aligned}$$

$$= L(\alpha)^*.$$

The null string ϵ always matches α^*, since ϵ is a concatenation of zero strings, all of which (vacuously) match α.

- x matches α^+ if x can be expressed as a concatenation of one or more strings, all of which match α:

$$\begin{aligned} L(\alpha^+) &= \{x_1 x_2 \cdots x_n \mid n \geq 1 \text{ and } x_i \in L(\alpha),\ 1 \leq i \leq n\} \\ &= L(\alpha)^1 \cup L(\alpha)^2 \cup L(\alpha)^3 \cup \cdots \\ &= L(\alpha)^+. \end{aligned}$$

Note that patterns are just certain strings of symbols over the alphabet

$$\Sigma \cup \{\epsilon, \emptyset, \#, @, +, \cap, \sim, {}^*, {}^+, (,)\}.$$

Note also that the meanings of $\#$, $@$, and $\sim$ depend on Σ. For example, if $\Sigma = \{a, b, c\}$ then $L(\#) = \{a, b, c\}$, but if $\Sigma = \{a\}$ then $L(\#) = \{a\}$.

Example 7.1

- $\Sigma^* = L(@) = L(\#^*)$.
- Singleton sets: if $x \in \Sigma^*$, then x itself is a pattern and is matched only by the string x; i.e., $\{x\} = L(x)$.
- Finite sets: if $x_1, \ldots, x_m \in \Sigma^*$, then

$$\{x_1, x_2, \ldots, x_m\} = L(x_1 + x_2 + \cdots + x_m). \qquad \square$$

Note that we can write the last pattern $x_1 + x_2 + \cdots + x_m$ without parentheses, since the two patterns $(\alpha + \beta) + \gamma$ and $\alpha + (\beta + \gamma)$ are matched by the same set of strings; i.e.,

$$L((\alpha + \beta) + \gamma) = L(\alpha + (\beta + \gamma)).$$

Mathematically speaking, the operator $+$ is *associative*. The concatenation operator $\cdot$ is associative, too. Hence we can also unambiguously write $\alpha\beta\gamma$ without parentheses.

Example 7.2

- strings containing at least three occurrences of a:

$$@a@a@a@;$$

- strings containing an a followed later by a b; that is, strings of the form $xaybz$ for some x, y, z:

$$@a@b@;$$

- all single letters except a:

$$\# \cap \sim a;$$

- strings with no occurrence of the letter a:

  $$(\# \cap \sim a)^*;$$

- strings in which every occurrence of a is followed sometime later by an occurrence of b; in other words, strings in which there are either no occurrences of a, or there is an occurrence of b followed by no occurrence of a; for example, aab matches but bba doesn't:

  $$(\# \cap \sim a)^* + @b(\# \cap \sim a)^*.$$

  If the alphabet is $\{a, b\}$, then this takes a much simpler form:

  $$\epsilon + @b.$$ □

Before we go too much further, there is a subtlety that needs to be mentioned. Note the slight difference in appearance between ϵ and ϵ and between $\emptyset$ and $\varnothing$. The objects ϵ and $\emptyset$ are *symbols* in the language of patterns, whereas ϵ and $\varnothing$ are *metasymbols* that we are using to name the null string and the empty set, respectively. These are different sorts of things: ϵ and $\emptyset$ are symbols, that is, strings of length one, whereas ϵ is a string of length zero and $\varnothing$ isn't even a string.

We'll maintain the distinction for a few lectures until we get used to the idea, but at some point in the near future we'll drop the boldface and use ϵ and $\varnothing$ exclusively. We'll always be able to infer from context whether we mean the symbols or the metasymbols. This is a little more convenient and conforms to standard usage, but bear in mind that they are still different things.

While we're on the subject of abuse of notation, we should also mention that very often you will see things like $x \in a^*b^*$ in texts and articles. Strictly speaking, one should write $x \in L(a^*b^*)$, since a^*b^* is a pattern, not a set of strings. But as long as you know what you really mean and can stand the guilt, it is okay to write $x \in a^*b^*$.