1 Closure Properties for P

The class P is closed under union, intersection, concatenation, and *. We just show closure under concatenation and *.

Frankly, the only one that is interesting is * since the others are rather easy.

Theorem 1. Let \( L_1, L_2 \in P \). Then \( L_1 L_2 \in P \).

Proof. Let TM \( M_1 \) decide \( L_1 \) in time \( p_1(n) \) (a polynomial) and TM \( M_2 \) decide \( L_2 \) in time \( p_2(n) \) (a polynomial). Here is the code for determining if a string \( x \in L_1 L_2 \).

1. Input string \( x \) of length \( n \).
2. Look at all \( n + 1 \) ways to split \( x \) into substrings \( y \) and \( z \), where \( x = yz \).
3. If \( y \in L_1 \) (run \( M_1 \) on \( y \)) and \( z \in L_2 \) (run \( M_2 \) on \( z \)) for some splitting of \( x \), then output TRUE. Else, output FALSE.

How fast is this algorithm? We run \( M_1 \) on strings of length 0, 1, 2, \ldots, \( n \) and \( M_2 \) on strings of length 0, 1, 2, \ldots, \( n \). (The string of length 0 is the empty string: note that if \( e \in L_1 \) and \( x \in L_2 \) then \( x \in L_1 L_2 \).) We use O-notation to avoid having to deal with details and constants. The run time is bounded above by

\[
O(p_1(0) + \cdots + p_1(n) + p_2(0) + \cdots + p_2(n)) \leq O(np_1(n) + np_2(n)).
\]

Since \( p_1 \) and \( p_2 \) are polynomials, \( np_1(n) + np_2(n) \) is a polynomial.

Theorem 1 is an illustration of why poly time is a good notion mathematically. Polynomials are closed under many operations (e.g., addition, multiplication), hence P is closed under many operations (e.g., concatenation). Classes like \( DTIME(n) \) and even \( DTIME(O(n)) \) are thought to not be closed under concatenation and many other operations. (We do not know if they are.)

Theorem 2. Let \( L \in P \). Then \( L^* \in P \).

Proof. Let TM \( M \) decide \( L \) in time \( p(n) \) (a polynomial).

Given \( x \) of length \( n \) we want to know if \( x \in L^* \). We could look at every way to break \( x \) up into substrings. That would not give a poly time algorithm since there are lots of ways to break up \( x \) (exercise: how many?).

We will actually solve a “harder” problem: given \( x \) of length \( n \), determine for ALL prefixes of \( x \), are they in \( L^* \). This is helpful since when we are trying to determine if, say,

\[ x_1 \cdots x_i \in L^* \]

we already know the answers to

\[ e \in L^* \]
\[ x_1 \in L^* \]
\[ x_1 x_2 \in L^* \]
\[ \vdots \]
\[ x_1 x_2 \cdots x_{i-1} \in L^*. \]

Intuition: \( x_1 \cdots x_i \in L^* \) IFF it can be broken into TWO pieces, the first one in \( L^* \), and the second in \( L \).

We now present the algorithm that will determine if \( x \in L^* \). The array \( A[i] \) will store if \( x_1 \cdots x_i \) is in \( L^* \).

```plaintext
input x of length n
A[0] = TRUE
for i = 1 to n do
    for j = 0 to n-1 do
        # Use machine M to test for membership in L
        if A[j] and (x_j, ..., x_{i-1}) in L then
            A[i] = TRUE
```

Theorem 2 is an illustration of why poly time is a good notion mathematically. Polynomials are closed under many operations (e.g., addition, multiplication), hence P is closed under many operations (e.g., concatenation). Classes like \( DTIME(n) \) and even \( DTIME(O(n)) \) are thought to not be closed under concatenation and many other operations. (We do not know if they are.)
What is the runtime of the above algorithm? The only time that matters is the calls to \(M\). There are \(O(n^2)\) calls to \(M\), all on inputs of length \(\leq n\), hence the runtime is bounded by \(O(n^2p(n))\). Since \(p(n)\) is a polynomial, \(n^2p(n)\) is a polynomial.

## 2 Closure Properties for NP

The class NP is closed under union, intersection, concatenation, and \(*\). We just show closure under concatenation. Frankly, all of these are easy.

**Theorem 3.** Let \(L_1, L_2 \in NP\). Then \(L_1L_2 \in NP\).

**Proof.** Since \(L_1 \in NP\) there exists set \(A_1\) in poly time \(q_1(n)\) and a poly \(p_1(n)\) such that
\[
L_1 = \{ x \mid (\exists y)[|y| = p_1(|x|) \land (x, y) \in A_1] \}
\]
Since \(L_2 \in NP\) there exists set \(A_2\) in poly time \(q_2(n)\) and a poly \(p_2(n)\) such that
\[
L_2 = \{ x \mid (\exists y)[|y| = p_2(|x|) \land (x, y) \in A_2] \}
\]
Given \(x\) we want to know if \(x \in L_1L_2\). Actually NO- we want evidence to VERIFY that \(x \in L_1L_2\). So we just need to know where the split happens and the corresponding \(y_1, y_2\).

(NOTATION: below we use \(x_1, x_2\). They are NOT the first two characters of \(x\). They are strings.)

\[
L_1L_2 = \{ x \mid (\exists x_1, x_2, y_1, y_2)[
\begin{align*}
\bullet & \ x = x_1x_2 \\
\bullet & \ |y_1| = p_1(|x_1|) \land (x_1, y_1) \in A_1 \\
\bullet & \ |y_2| = p_2(|x_2|) \land (x_2, y_2) \in A_2
\end{align*}
\}
\]

Notice that
\[
|x_1, x_2, y_1, y_2| \leq O(n + n + p_1(n) + p_2(n))
\]
which is a poly in \(n\). So the witness is short.

Notice that testing \((x_1, y_1) \in A_1\) and \((x_2, y_2) \in A_2\) takes times bounded by
\[
O(q_1(n + p_1(n)) + q_2(n + p_2(n)))
\]
which is a polynomial.

\[\square\]