Bram Schuijff
CSCI 387, §S01/02
Homework 8
April 22nd, 2025 by 10:30am

1. Let A be a language that is PSPACE-hard. This means that for any B in PSPACE, it may be reduced in polynomial time to A. Then, we want to show that for any C in NP, C can be reduced in polynomial time to A. Let C be in NP.

2. To demonstrate that A is in L, we can produce a logarithmic space decider for A. We define such a decider D as follows:

   1. On input $\langle w \rangle$:
   2. Initialize a counter that is set to 0.
   3. While scanning from left to right across w:
   4. If we encounter a left parenthesis, increment the counter by 1.
   5. Else if we encounter a right parenthesis:
   6. If the counter is at 0, reject.
   7. Else, decrement the counter by 1.
   8. If the counter is not equal to 0 by the time we read all of w, reject. Otherwise, accept.

   I claim that D decides A. This is because if w is a string of properly-nested parentheses, then for each left parenthesis, we increment our counter by 1 and for each right parenthesis, we subsequently decrement our counter by 1. Therefore, if our counter is greater than 0 by the end of w, there is an unclosed left parenthesis and we reject. Meanwhile, if our counter is at 0 and we encounter a right parenthesis, this means that right parenthesis will remain unclosed no matter what comes after it in w and so we reject. Meanwhile, if our counter is equal to 0 by the end of w and we have not rejected, all left parentheses are closed and there are no outstanding right parentheses. In this case, we have properly-nested parentheses and we accept. Therefore, D decides A.

   To demonstrate that D takes log space, we notice that D simply maintains a counter. We have discussed previously that counters that count occurrences of symbols in the input tape is logarithmic space with regards to the input. Therefore, D is log space. Therefore, because D decides A and D is logarithmic space, A is in L.

3. To show that A is in L, we can produce a logarithmic space decider for A. We define such a decider D as follows:

   1. On input $x\#w$:
   2. Maintain 4 counters l_x, l_w, c_1, c_2 each initially set to 0.
   3. For each character in x:
   4. Increment l_x by 1.
   5. For each character in w:
   6. Increment l_w by 1.
   7. If $l_x > l_w$, reject.

8. While $c_1 < l_w - l_x + 1$:
9. Scroll back to the #.
10. While $c_2 < l_x$:
11. Taking Pythonic string indexing, if $x[c_2]$ is not equal to $w[c_1 + c_2]$:
12. Break to the outer loop.
13. Increment c_2 by 1.
14. If c_2 is equal to l_x, accept.
15. Else, set $c_2 = 0$ and increment c_1 by 1.
16. Reject.

I claim that D decides A. This is because for any x and w, we start by counting up the lengths of w and x and storing them in l_w and l_x respectively. Then, if $l_x > l_w$, we reject because it cannot be that x is a substring of w if x is longer than w. Then, for each starting index c_1 in w, we check whether the l_x characters after and including c_1 are equal to the characters of x in order. If they are, then w contains x as a substring at starting index c_1. Meanwhile, if we make it to the end of w and have not yet found a starting index in w whereby the l_x characters after and including that index are equal to x, x is not a substring of w and so we reject. Therefore, D decides A.

To prove that D is logarithmic space, we find that we maintain a constant number of counters which are each bounded by $|x|$ or $|w|$. Because maintaining a counter with magnitude in the length of the input string is logarithmic with respect to the input string and we have a constant number of these counters, we find that D occupies logarithmic space. Therefore, because D decides A and D occupies logarithmic space, A is in L.

4. To demonstrate that A_{NFA} is NL-complete, we first need to show that A_{NFA} is in NL. To do this, we produce a log space verifier V for A_{NFA} with x as the witness. I will suppose that x contains a computation history of M on w. This computation history can be represented as the current state of M followed by the remaining unread tape symbols. As such, $x[i]$ will denote the i-th state that M got to when run on w.

1. On input $\langle M, w, x \rangle$:
2. Maintain four counters l_w, l_x, r_w, p_x that are initially set to 0.
3. For each symbol in w:
4. Increment l_w by 1.
5. For each symbol in x:
6. Increment l_x by 1.
7. Scroll back to the first symbol on the input tape.
8. If the first state in x is not the start state, reject.
9. While $p_x < l_x - 1$:
10. Initialize a temporary variable $t = 0$.
11. For each symbol on the computation history's input tape at $x[p_x + 1]$:

        Increment t by 1.
    If $t = r_w$:
        If $x[p_x + 1]$ cannot be reached from $x[p_x]$ by an ϵ transition, reject.
    Else if $t = r_w - 1$:
        If $x[p_x + 1]$ cannot be reached from $x[p_x]$ by reading the current symbol, reject.
    Clear t from the tape.
If $x[p_x]$ is an accepting state, accept.