Домашняя работа 5 — Кузнецов

1 Задание 5

1. C37. f, g можно посчитать с использованием $O(\log n)$ памяти. Тогда и $f \circ g$ можно посчитать с использованием $O(\log n)$ памяти.

   Основная трудность здесь в том, что мы не можем решить эту задачу тривиальным образом: вычислить функцию g, вывести результат на рабочей ленте, а потом запустить на этом результате как на входе вычисление функции f. Дело в том, что результат работы функции g может иметь размер больше, чем $O(\log n)$. Идея решения состоит в том, что мы будем вычислять результат функции g по одному символу, передавать на вход функции f и сразу стирать (иногда нам придётся возвращаться к символам, которые функция g вывела раньше, для этого нам придётся перезапускать машину для функции g).

   Теперь опишем решение более детально. Машина Тьюринга M_{fg} для
   $f \circ g$ будет иметь входную ленту, на которой записан вход x, выходную ленту, рабочие ленты машины для f и машины для g, вспомогательную рабочую ленту A, на которой мы будем писать вывод функции g по одному символу (и сразу стирать), ленту B для запоминания состояния машины Тьюринга для g. Машина M_{fg} будет работать следующим образом: мы запускаем машину Тьюринга для g и ждём, пока она выведет один символ в первой ячейке ленты A. Когда этот символ выводится, мы передаём его функции f. Если машина Тьюринга для f принимает решение пойти вправо на ленте A, то мы ждём пока машина Тьюринга для g выведет следующий символ на ленте A (мы выводим его в той же первой ячейке ленты A) и передаём его машине Тьюринга для f, и т.д. Если же в некоторый момент машина Тьюринга для f принимает решение пойти влево на ленте A, то нам нужно восстановить предыдущий символ, который был выведен машиной Тьюринга для g

на ленте A. Для этого мы запоминаем на ленте B конфигурацию машины Тьюринга для g (то есть положения её головок на рабочих лентах, положение головки на входе x длины n — для этого надо $\log n$ битов, и текущее состояние). После этого перезапускаем машину для g и ждём, пока следующим состоянием не будет наше — этот-то момент нам и нужен, в этот момент нам и интересно, какой символ машина выводит его и подаём на вход f. Если же машина g в этот предыдущий момент никакого символа не выводит, а выводила символ до этого, то ещё раз перезапускаем и находи предыдущее состояние, и т.д. пока не дойдём до состояния с выводом символа. И т.д. Когда машина для f будет идти вправо, мы будем продолжать работу машины для g, когда будет идти влево — будем перезапускать машину для g и находить последнее предшествующее состояние, в котором был вывод какого-то символа. Вот и всё.

Замечание 1.1. *Остаётся один пробел, который, кажется, был и в рассуждении на лекции. Мы запоминаем место на входной ленте, на котором находилась головка машины, и говорим, что для этого требуется $O(\log n)$ битов. Но это так, если мы уверены, что головка машины не вышла за пределы входа. А что, если она дошла до последнего символа входа и пошла дальше вправо?*

Однако, этот пробел можно заполнить. Допустим, у нас есть k рабочих лент, на каждой мы используем памяти не более $K \log n$, и S — число состояний нашей машины Тьюринга. Тогда, если головка отходит вправо от последнего символа входа более чем на $S(K\log n)^k 2^{kK\log n}$, то у нас во время "экскурсии" головки вправо (когда она не возвращалась во входное слово x) встретятся два разных момента, в которые и состояние машины было тем же самым, и слова на рабочих лентах, и положения головок на них совпадали. А тогда наша машина будет и дальше уходить вправо, и никогда не остановится — она, в некотором смысле, зациклилась. А такого быть не должно, машина должна завершать работу.

2. C38. а) Если $A \le_l B$, то $A \le_p B$. Функция f, выполняющая сведение, лежит в L. На лекции было доказано (на 41-ой минуте), что $L \subseteq P$. Значит, f лежит в P. Значит, наше сведение является и сведением по Карпу.

   б) Если $A \le_l B$ и $B \le_l C$, то $A \le_l C$. Пусть функция, выполняющая первое сведение, g, а второе f. Тогда $f \circ g$ выполняет сведение A к C, а она по задаче C37 лежит в L.

в) Если $A \le_l B$ и $B \in L$, то $A \in L$. Пусть функция, выполняющая сведение $A \le_l B$, есть g, а решающая B — f. Тогда $f \circ g$ решает A, и по С37 она лежит в L.

3. С39. а) Доказано на лекции.

4. С33. После такой подсказки задача становится намного проще!

   Мы можем закодировать все 3-SAT натуральными числами (быть может, некоторым числам не будет соответствовать никакая 3-SAT; то есть, мы записываем 3-SAT как слова в алфавите из 0 и 1, а если прочитать это как двоичную запись некоторого числа, то это и будет кодировка 3-SAT натуральным числом; договоримся, что запись 3-SAT всегда начинается с 1, — чтобы не получить, что, например, 00101 и 101 соответствуют одному и тому же числу).

   Будем строить язык A таким образом. Слово $x0$ лежит в A, если и только если $3-SAT$, соответствующая слову x, выполнима (если слово x не соответствует никакой $3-SAT$, то $x0$ не леджит в A). Принадлежность слов $x1$ языку A будем определять следующим образом: $x1$ принадлежит языку A тогда и только тогда, когда x принадлежит языку B, где язык B построен ниже так по аналогии с доказательством теоремы Бейкера-Гила-Соловея (я беру доказательство, изложенное в книге Arora, Barak), так, что
   $U_B \notin P^A$, где $U_B = \{1^k \mid \exists x \in B, |x| = k\}$.

   Покажем, что язык $U_A = \{1^k \mid \exists z \in A\colon z = x1, |x| = k\} = U_B$ содержится в $NP^A \setminus P^A$. Почему U_A содержится в NP^A? Потому что принадлежность слова 1^k языку U_A мы можем определить с помощью подсказки полиномиальной длины — в роли этой подсказки выступает соответствующее слово z ($z = x1, |x| = k$), и мы просто проверяем принадлежность слова x языку A с помощью оракула A.

   Почему U_A не содержится в P^A? Потому что $U_A = U_B$.

   Покажем, что если U_A сводится к $3-SAT$, то $U_A \in P^A$. Действительно, допустим, мы построили полиномиальное сведение (с оракулом A) U_A к $3-SAT$, то есть по слову построили $3-SAT$. Тогда с помощью оракула A мы можем определить, выполнима ли построенная $3-SAT$ (если соответствующая ей строка кодируется как x, мы запрашиваем у оракула, лежит ли $x0$ в A). Таким образом, мы полиномиально с использованием оракула A выясняем, лежит ли слово в языке U_A. Поэтому в этом случае $U_A \in P^A$. Противоречие.

Осталось построить язык B, такой, что $U_B \notin P^A$. Будем строить его по аналогии с доказательством теоремы Бейкера-Гила-Соловея. Занумеруем все машины Тьюринга с оракулом A, так что M_i — iя машина Тьюринга с оракулом A, и каждая машина Тьюринга встречается в такой нумерации бесконечно много раз. Теперь будем последовательно определять принадлежность строк x языку B (при этом одновременно будет определяться принадлежность слов вида $x1$ языку A).

Итак, вначале, на нулевом этапе, язык B пустой.

Опишем i-ый этап. На этом этапе мы уже определили принадлежность конечного числа строк языку B. Возьмём достаточно большое n, такое, что длины всех этих строк меньше n. Относительно всех строк длины меньше n с неопределённым статусом решаем, что они не лежат в B. Теперь запускаем машину M_i на входе 1^n на $2^n/10$ шагов. Когда эта машина обращается к оракулу A и спрашивает о принадлежности строк вида $x0$ к языку A, мы отвечаем так, как определили выше принадлежность таких строк языку A. Когда же эта машина спрашивает о принадлежности строк вида $x1$ языку A, если принадлежность x языку B уже определена, мы отвечаем соответственно. Если же принадлежность строки x языку B ещё не определена, мы говорим, что $x1$ не принадлежит языку A, и решаем, что x не принадлежит языку B. Заметим, что до этого момента мы ни относительно одной строки длины n не решили, что она принадлежит языку B (хотя, возможно, относительно некоторых строк длины n решили, что они не лежат в B). Теперь делаем так, что если машина M_i останавливается за $2^n/10$ шагов, то её ответ на строке длины 1^n не совпадает с принадледжностью этой строки языку B. Если машина M_i принимает вход 1^n, мы решаем, что все строки длины n не лежат в B (и тогда 1^n не лежит в U_B). Если M_i принимает строку 1^n, мы выбираем строку x длины n, такую, что о принадлежности строки $x1$ мы не спрашивали оракул A (такая строка найдётся, потому что всего запросов к оракулу было не более $2^n/10$), и решаем, что эта строка лежит в B, а тогда $1^n \in U_B$. Итак, в любом случае ответ M_i не совпадает с принадлежностью строки 1^n языку U_B. Отсюда заключаем, что $U_B \notin P^A$.