Задание 3, задача 1.

Красильников Иван

12 апреля 2010 г.

Прежде всего нужно проверить, что у автоматов одинаковый размер алфавита. Если это не так, то они не эквиваленты (потому что по условию, у каждого автомата из каждой вершины есть ребра по всем символам).

Рассмотрим два автомата как один, «объединенный» автомат, со множеством состояний, равным объединению множеств состояний двух автоматов, и с двумя начальными состояниями s_0, s_1, равным начальным состояниями каждого из двух исходных автоматов.

Пусть $L(v)$ – множество строк, которые принимает этот автомат, если бы начальным состоянием было v. Назовем две вершины x и y эквивалентными, если $L(x) = L(y)$. Очевидно, что это отношение эквивалентности (транзитивно и т.п.), и таким образом разбивает вершины на классы эквивалентности. Задача состоит в том, чтобы проверить, эквивалентны ли s_0 и s_1.

Будем это проверять так. Сначала сделаем предположение (гипотезу), что s_0 и s_1 действительно эквивалентны. Дальше определим, эквивалентность каких других пар вершин следует из этого предположения: из эквивалентности двух вершин x и y следует эквивалентность $f(x, c)$ и $f(y, c)$, где f – функция перехода, c – любой символ алфавита. В конце проверим, не привело ли это к противоречию: если какая-то терминальная вершина ($\varepsilon \in L(x)$) оказалась эквивалентна нетерминальной ($\varepsilon \notin L(x)$), то исходное предположение было неверно, и автоматы неэквивалентны.

А иначе они действительно эквиваленты: полученные в ходе работы алгоритма отношения эквивалентности позволяют нам создать новый автомат, вершины которого – классы эквивалентности, ребро между двумя вершинами по символу c существует если есть ребро между двумя вершиными внутри соответствующих классов по этому символу. Это будет детерминированный автомат (из класса x не могут существовать ребра в разные классы y и z по одному символу, иначе бы y и z на предыдущем шаге были бы признаны эквивалентными). Очевидно, что оба входных автомата будут эквиваленты этому только что построенному автомату, а значит и эквивалентны между собой.

Сложность по времени.

Я использую стандартную структуру union-find со всеми нужными эвристиками для поддержания системы классов эквивалентности. Будем считать для простоты, что она умеет делать union и find за $O(1)$, хотя это и не совсем так.

Самый затратный шаг в алгоритме – найти все отношения эквивалентности, вытекающие из предположения, что s_0 эквивалентно s_1. Шаг реализован следующим образом:

```
do {
  X = пустой ассоциативный массив, отображающий
             пары (вершина, символ) в списки вершин.

  для каждого ребра e, задающего переход из x в y по символу c:
     добавить y в X[Find(x), c]

  для каждого списка L в X:
     слить (операцией Union) вершины этого списка
} while (произошло хотя бы одно слияние);
```

Сложность одной итерации цикла – линейная по размеру входу (число вершин $\times$ размер алфавита), число итераций ограничено сверху числом вершин минус 1 – слияний не может быть больше этого числа.

Итого сложность: $O(n^2 l)$, где n – общее число вершин в автоматах, l – размер алфавита.