Домашнее задание №2

Д.А. Першин

25 марта 2015 г.

1 Словесное описание алгоритма

Даны два детерминированных конечных автомата A и B. Необходимо определить, эквивалентны ли они. Для этого построим автомат, состояния которого равны объединению состояний двух автоматов A и B. Далее проверим эквивалентность начальных состояний автоматов A и B в объедененном автомате. Если состояния эквивалентны (неразличимы), то автоматы эквивалентны, иначе неэквивалентны. Эквивалентность состояний будем проверять с помощью алгоритма заполнения таболицы неэквивалентности.

Алгоритм заполнения таболицы неэквивалентности:
Состоянием $\delta(s, w)$ назовем состояние, в которое автомат переходит из состояния s по цепочке w. Два состояния s_1 и s_2 являются эквивалентными, если для всех входных цепочек w состояние $\delta(s_1, w)$ является допускающим тогда и только тогда, когда состояние $\delta(s_2, w)$ - допускающее.
Таблица неэквивалентности это таблица размера $n \times n$, где n - кол-во состояний в автомате, на перечечении строки i и столбца j которой находится 1, если состояния неэквиваленты (различимы), иначе 0. Заполнять таблицу будем следующим образом: во первых различимыми являются пары состояний, одно их которых является терминальным (допустимым), а другое нет, отметим их. Далее различимыми являются состяния, которые переходят в различимые состояния по одним и тем же символям алфавита. Для этого построим список обратных ребер (переходов) для каждого состояния и будем двигаться от уже различимых состояний по обратным ребрам, отмечая их тоже как различимые. Процесс будем проводить итеративно, используя некоторый аналог поиска в ширину (добовляя вновь полученные различимые состояния в очередь и отмечая все состояния, достижимые из текущего состояния по обратным ребрам).

Кратко опишем алгоритм проверки эквивалентности автоматов:

1. построим автомат $C = A + B$.

2. построим таблицу неэквивалентности для автомата C.

3. определим различимы ли начальные состояния автоматов A и B в объедененном автомате C.
   - если состояния различимы, то автоматы не эквивалентны.
   - если состояния неразличимы, то автоматы эквивалентны.

Алгоритм:

- Пусть a - начальное состояние автомата A, b - начальное состояние B, M - таблица неэквивалентности объедененного автомата C (алгоритм 2). Определим являются ли автоматы эквивалентными:

  Algorithm 1: проверка эквивалентности автоматов

  procedure IsEquivalent(A, B)
      $C = A + B$ ▷ C - объединение автоматов A и B
      $M = NonequivalenceTable(C)$
      if $M[a][b] = true$ **then** ▷ если состояния различимы
          return false ▷ автоматы A и B неэквивалентны
      else
          return true ▷ автоматы A и B эквиваленины
      end if
  end procedure

- Создадим пустую очередь из пар неэквивалентных состояний Q, M - искомая таблица неэквивалентности, $\delta^{-1}(i, j)$ - список обратных ребер (алгоритм 3), T - список терминальных вершин автомата C, Σ - алфавит. Построим таблицу неэквиватентности:

  Algorithm 2: построение таблицы неэквивалентности

  procedure NonequivalenceTable(C)
      for $\forall i \in [0, n-1]$ **do**
          for $\forall j \in [0, n-1]$ **do**
              if $M[i][j] = false$ and $T[i] \neq T[j]$ **then** ▷ одно состояние терминальное, другое нет
                  $M[i][j] = M[j][i] = true$ ▷ пометить состояния как неэквив.
                  $Q.push(< i, j >)$ ▷ добавить пару вершин в очередь
              end if
          end for
      end for

      $\delta^{-1} = InverseEdges(C)$

      while $!Q.empty()$ **do** ▷ помечаем состояния, достижимые из различимых состояний по обратным ребрам как различимые
          $< u, v >= Q.poll()$

```
        for c ∈ Σ do
            for r ∈ δ^{-1}[u][c] do
                for  s ∈ δ^{-1}[v][c] do
                    if M[r][s] == false then
                        M[r][s] = M[s][r] = true
                        Q.push(< r, s >)
                    end if
                end for
            end for
        end for
    end while

    return M
end procedure
```

- Обозначим через C автомат, через $T(s, t)$ матрицу переходов автомата (переход из состояния s по символу t), $\delta^{-1}(s, t)$ - искомый список обратных ребер для состояния s по символу t. Построим список обратных ребер:

Algorithm 3: построение списков обратных ребер

```
procedure InverseEdges(C)
    for ∀s ∈ C do                               ▷ для всех состояний автомата C
        for ∀t ∈ s do                           ▷ для всех переходов из состояния s
            if T(s, t) ≠ None then              ▷ если переход существует
                δ^{-1}(T(s, t), t)).push(s)     ▷ добавить обратное ребро
            end if
        end for
    end for

    return δ^{-1}
end procedure
```

2 Доказательство корректнсти

Корректность алгоритмов объединения автоматов и построения списков обратных ребер достаточно очивидна. Докажем корректность алгоритма построения таблицы неэквивалентности. Доказательство будем проводить по индукции. Пусть s_1 и s_2 - состояния, для которых существует символ c, приводящий их в различимые состояния $s'_1 = \delta(s_1, c)$ и $s'_2 = \delta(s_2, c)$. Тогда существует цепочка w, различающая их, т.е существуют различимые состояния $\delta'(s'_1, w)$ и $\delta'(s'_2, w)$, одно из которых является допускающим, а другое нет. Но в этом случае цепочка cw отличает состояния s_1 и s_2, т.к. $\delta(s_1, cw)$ и $\delta(s_2, cw)$ - различимы, следовательно s_1 и s_2 также различимы.

Теперь докажем, что если начальные состояния автоматов в объединенном автомате различимы, то автоматы неэквивалентны. Предположим, что начальные состояния оказались различимы. В этом случае существует входная цепочка w, которая переводит одина автомат в допустимое состояние, а другой автомат - в недопустимаое, но это и является критерием неэквиватентности автоматов. Если же состояния неразличимы, то по любой входной цепочке оба автомата переходят либо в допустимое состояние, либо в недопустипое, а это и есть критерий их эквивалентности.

3 Асимптотические оценки

Пусть суммарное кол-во состояний в автоматах A и B равно n, длина алфавита Σ (по условию $\Sigma \leq 26$). В результате получаем сложность по памяти $O(n^2)$, так как мы используем матрицу переходов размера $n \times \Sigma$, вектор терминальных сосотояний длины n, таблицу неэквивалентности размера $n \times n$, очередь неэквивалентных состояний, максимальная длина которой не превышает n и списки обратных ребер максимальной длины, равной кол-ву ребер $(n \times \Sigma)$.

Сложность по времени равна $O(n^2)$, так как для объединения автоматов требуеся линейное время $O(n)$, для построения списков обратных ребер требуется $O(n \times \Sigma)$ (необходимо перебрать все ребра), а для построения таблицы неэквивалентности требуется $O(n^2)$ ($O(n^2)$ в первом цикле для пометки и добавления в очередь всех пар состояний, одно из которых терминальное, а другая нет, и $O(n \times \Sigma)$ для обхода оставшихся состояний по спискам обратных ребер).

Итого, получаем временную сложность $O(n^2)$, сложность по памяти $O(n^2)$