Проект

Бондаренко Александр

23 октября 2021 г.

Мой язык описания ДКА немного изменился: теперь все состояния имеют такие названия: **qi**, где i– индекс состояния(нумеруются с нуля).

Моей задачей было распарсить язык, на котором описан мой автомат и, используя алгоритм Хопкрофта, минимизировать ДКА. Алгоритм работает за время $\mathcal{O}(|\Sigma| \cdot |Q| \cdot \log |Q|)$.

Запустив *bash*-скрипт таким образом: `./run.sh input.txt`, мы получим структуру нашего автомата в файле `input.txt.out`, а также его **минимизированную** версию.

Также был написан стресс–тест(скрипт `stress_test.py`), который показал, что алгоритм на автомате с алфавитом **размера 1** и с 10^5 **состояниями** отработает за 0.7 секунд. Если наш алфавит– это вся таблица ASCII, то алгоритм за 2 секунды сможет минимизировать автомат с 10^4 состояниями. Так можно запустить стресс тест: `./run_stress.sh input.txt`.

Есть и другие алгоритмы минимизации. К примеру, алгоритм Мура, который в среднем работает $\mathcal{O}(n \log n \log n) - -\mathcal{O}(n \log n)$ времени. Но есть тесты, на которых будет $\mathcal{O}(n^2)$, как у самого простого алгоритма минимизации.

Кстати, задача минимизации НКА принадлежит классу $PSPACE$. Тогда в предположении, что $PSPACE \neq P$, полиномиального алгоритма для этой задачи не существует.

Источники, которые помогли с эффективной реализацией алгоритма и его пониманием:
тык1
тык2
тык3