

Детерминизация конечного автомата

Кузнецов А.Д.

Оглавление

0.1	Краткая теоретическая часть	1
0.1.1	Автомат	2
0.1.2	Теорема Клини	2
0.2	Постановка задачи	3
0.3	Алгоритм детерминизации автомата	3
0.3.1	Удаление λ -переходов	3
0.3.2	Детерминизация КНА	3
0.3.3	Оценка сложности алгоритма	6
0.4	Реализация алгоритма детерминизации автомата	6
0.4.1	Кодирование автоматов (реализация классов)	6
0.4.2	Реализация алгоритма удаления λ -переходов	6
0.4.3	Реализация алгоритма детерминизации КНА	6
0.4.4	Уязвимые для критики места	6
0.5	Тестирование	6
0.5.1	Unit-тестирование	6
0.5.2	Умное тестирование	6
0.6	Использование алгоритма	6
0.6.1	Формат файла-автомата (.fsa)	6
0.6.2	Компиляция и запуск основной программы	6
0.7	Заключение	6

0.1 Краткая теоретическая часть

Ниже приводятся определения и утверждения, которые будут использованы для дальнейших пояснений к реализации алгоритма детерминизации.

0.1.1 Автомат

Конечный недетерминированный автомат (КНА) M — это кортеж вида

$$M = (A, Q, q_0, F, \delta),$$

где

- $A = \{a_0, a_1, \dots, a_{m-1}\}$ — входной алфавит, т.е. множество символов, причем конечное: $|A| = m$;
- $Q = \{q_0, q_1, \dots, q_{n-1}\}$ — множество состояний автомата, тоже конечное: $|Q| = n$;
- q_0 — начальное состояние автомата, то есть $q_0 \in Q$;
- $F \subset Q$ — выходные состояния автомата;
- $\delta : Q \times A \rightarrow 2^Q$ — функция переходов автомата.

Если в автомате разрешены переходы по пустому символу, то входной алфавит дополняется фиктивным символом λ (в иностранной литературе его чаще обозначают ε). В таком случае автомат называют λ -КНА (ε -FSA). Входной алфавит при этом будем обозначать так же, то есть если речь идет о λ -КНА, то подразумевается, что $A \leftarrow A \cup \{\lambda\}$, а мощность $|A| = m + 1$.

Конечный детерминированный автомат (КДА) M — это такой КНА, где функция переходов δ выглядит так:

$$\delta : Q \times A \rightarrow 2^Q,$$

т.е. из любого состояния по любой букве возможен переход в точности в одно состояние — это и обеспечивает детерминированность работы такого автомата.

0.1.2 Теорема Клини

Пусть $A = \{a_0, \dots, a_{n-1}\}$ — произвольный алфавит. Язык $L \subseteq A^*$ является элементом полукольца регулярных языков $R(A)$ в алфавите A тогда и только тогда, когда он допускается некоторым конечным автоматом.

0.2 Постановка задачи

Теперь сформулируем задачу в вышеуказанных терминах:

Реализовать алгоритм преобразования λ -КНА M в КНА \hat{M} так, чтобы распознаваемые ими языки совпадали, т.е. $L(M) = L(\hat{M})$.

0.3 Алгоритм детерминизации автомата

Алгоритм детерминизации осуществляется в два шага - удаление λ -переходов (то есть переход от λ -КНА к КНА) и непосредственно детерминизация КНА ($\text{КНА} \rightarrow \text{КДА}$).

0.3.1 Удаление λ -переходов

Для любого λ -КНА можно построить эквивалентный ему КНА (Здесь и далее под эквивалентностью автоматов подразумевается совпадение распознаваемых ими языков). Доказательство этого утверждения входит в обоснование алгоритма детерминизации, который описан в учебниках [1]. Основная идея этого шага такая - из состояния q_i существует переход по букве a в состояние q_j тогда и только тогда, когда выполняется одно из двух условий:

- $q_j \in \delta(q_i, a)$;
- $\exists (q_1, q_2, \dots, q_p) \in \bigcup_{p=1}^n Q^p :$
 $(q_1 \in \delta(q_i, \lambda)) \wedge (\forall k = 1, p-1)(q_k \in \delta(q_{k+1}, \lambda)) \wedge (q_p \in \delta(q_i, a)).$

То есть, если из состояния q_i какая-то цепь пустых переходов ведет в состояние, из которого есть переход в q_i по букве a , то в новом КНА будет переход из q_i в q_i по букве a .

0.3.2 Детерминизация КНА

Теперь у нас есть КНА, который всегда можно детерминизировать (это утверждение тоже есть и доказано в учебнике).

Алгоритм интуитивно следует понимать так: если моделировать КНА на компьютере, то прежде всего придется реализовать автомат так, чтобы

он мог находиться одновременно в различных состояниях, в которые он может попасть по прочитанному подслову входной последовательности. Точно так же и тут - пусть в КДА состояния будут в биективном отношении с совокупностью подмножеств состояний входного КДА. Такие состояния результирующего КДА будем называть гиперсостоянием (исключительно ради удобства, в коде, реализующем алгоритм, они так и называются). Помимо этого, будем говорить, что состояние входит в гиперсостояние, если это состояние входит в то подмножество Q , в котором гиперсостояние находится в биективном отношении.

Теперь вполне разумна мысль, что переход из одного гиперсостояния в другое по букве возможен тогда, когда хотя бы из одного состояния в первом гиперсостоянии есть переход по этой букве в состояния из второго гиперсостояния. Вообще и обратное следствие верно, но доказывать его сложнее.

Так как уже невозможно это произносить, читать и, поверьте мне, печатать, перейдем непосредственно к алгоритму.

Вход: $M = (A, Q, q_0, F, \delta);$

Выход: $\hat{M} = (\hat{A}, S, s_0, F_s, \hat{\delta});$

- 1: $\hat{A} := A;$ // Алфавиты совпадают
- 2: $s_0 := \{q_0\}$ // Входное гиперсостояние соответствует входному состоянию
 исходного автомата
- 3: $F_s = \emptyset$ // Множество выходных гиперсостояний
- 4: $DET = \emptyset$ // Множество уже детерминизированных гиперсостояний
- 5: $NONDET = \{s_0\}$ // Множество еще недетерминизированных гиперсостояний
- 6: **пока** $NONDET \neq \emptyset$
- 7: $state := choice(NONDET);$ // Выбираем како-либо элемент из
 множества и извлекаем его
- 8: **для всех** $a \in A$
- 9: $next := \emptyset;$ // Новое гиперсостояние при переходе по букве a
- 10: **для всех** $q \in state$
- 11: $next := next \cup \delta(q, a);$
- 12: $\hat{\delta}(state, a) := next;$ // Добавляем соответствующий переход по
 букве
- 13: **если** $next \notin DET$ **то**
- 14: $NONDET := NONDET \cup \{next\};$ // Если гиперсостояние
 не было детерминированно, то добавляем его в множество ожидающих
 детерминизацию
- 15: **если** $state \cap F \neq \emptyset$ **то**
- 16: $F_s := F_s \cup \{state\};$ // Если хотя бы одно состояние гиперсостояния
 было выходным, то все гиперсостояние - выходное
- 17: $S := DET ;$
- 18: $\hat{M} = (\hat{A}, S, s_0, F_s, \hat{\delta}) ;$ // Возвращаем детерминированный автомат

0.3.3 Оценка сложности алгоритма

0.4 Реализация алгоритма детерминизации автомата

0.4.1 Кодирование автоматов (реализация классов)

0.4.2 Реализация алгоритма удаления λ -переходов

0.4.3 Реализация алгоритма детерминизации КНА

0.4.4 Уязвимые для критики места

0.5 Тестирование

0.5.1 Unit-тестирование

0.5.2 Умное тестирование

0.6 Использование алгоритма

0.6.1 Формат файла-автомата (.fsa)

0.6.2 Компиляция и запуск основной программы

0.7 Заключение

~~Реализовано все круто, добавить нечего, по-чаще бы так писали код.~~

Bibliography

- [1] А. И. Белоусов, С. Б. Ткачев. Дискретная математика.
Издательство МГТУ им. Н. Э. Баумана, Москва, 2002.