Примеры машин Тьюринга (МТ)

Краткое описание этой модели алгоритма содержится в учебнике «Дискретная математика» (вып. 19 серии «Математика в техническом университете») в дополнении Д.7.4).

1. Распознавание ровно одного вхождения слова aba, т.е. необходимо написать программу МТ, которая вычисляет функцию

$$f(x)=\begin{cases}1, & \text{если слово } aba \text{ входит в слово } x\in\{a,b\}^* \text{ ровно 1 раз}\\ 0 & \text{иначе}\end{cases}$$

Рабочий алфавит машины $\{a,b,0,1\}$, но в нем выделена входная $\{a,b\}$ и выходная {0, 1} части.

Программа на следующей странице.

Замечания к программе.

1) Вводится дополнительное состояние q_{00} как начальное состояние поиска в непустом слове. Это сделано, главным образом, из методических соображений, чтобы специально выделить случай пустого слова на входе. Но можно избежать этого и в программу записать команду $q_0\square\to q_9\square, L$, т. е. команду, инициирующую возврат головки при неуспешном поиске. Тогда в случае пустого входного слова головка оказывается в состоянии q_9 сразу на маркере начала ленты, и выполняется команда с левой частью $q_9 *$, т. е. делается сдвиг на одну ячейку вправо, в пустую ячейку записывается 0, и машина останавливается (с головкой на маркере).

2) Стандартно в программах МТ через q_0 обозначается начальное состояние, а через q_f заключительное. **Заключительное состояние не может присутствовать в левых частях команд**, т.е. переход в заключительное состояние при установке головки на маркер начала ленты означает нормальный останов машины.

3) Внутри входного слова никаких пробелов (пустых ячеек) не может быть. Первый пробел означает конец слова. В данной программе после завершения поиска слово при возврате головки стирается и в первую после маркера начала ленты ячейку записывается признак результата (1 при успехе и 0 при неуспехе).

4) В данной программе фигурируют параметры: α, пробегающий множество букв, r, пробегающий множество состояний, и i, пробегающий множество номеров состояний. В любом случае **параметр должен пробегать конечное множество,** так как иначе строка с параметром определяет бесконечное множество команд, что недопустимо.

5) Написанная ниже программа (как и все прочие) есть программа **детерминированной** МТ, т. е. в ней не должно быть двух (и более) различных команд с одинаковыми левыми частями (наличие таких команд приводит к тому, что можно назвать **конфликтом команд**; таких конфликтов в программах не должно быть). Можно заметить, что порядок команд в программе не важен. Команда выполняется по текущему состоянию и обозреваемому символу, которые в каждый момент времени определены однозначно.

6) Написанную ниже программу легко модифицировать для распознавания слов, в которых имеет место **не менее двух вхождений** слова aba. Изменятся лишь условия успешного и неуспешного поиска.

$q_0 * \to q_0 *, R$

$q_0 \square \to q_f\, 0, L$ / /если входное слово пустое,

то печатаем 0 и останавливаемся на маркере начала ленты

$q_0 a \to q_1 a, R$ / /буква a может быть первой буквой искомого слова;

переход в состояние q_1 означает ожидание второй буквы, т.е. буквы b

$q_0 b \to q_{00} b, R$ / /состояние q_{00} можно охарактеризовать, как

начальное состояние поиска в непустом слове

$q_{00} a \to q_1 a, R$

$q_{00} b \to q_{00} b, R$

$q_1 a \to q_1 a, R$ / /"зависаем" в состоянии q_1, ожидая букву b

$q_1 b \to q_2 b, R$ / / прочитан префикс ab

$q_2 a \to q_3 a, R$ / / найдено 1-е вхождение искомого слова, но нужно проверить

наличие 2-го вхождения и при этом надо учесть, что последняя буква 1-го

вхождения может оказаться первой буквой 2-го

$q_2 b \to q_{00} b, R$ / /прочитан префикс abb, поиск начинается заново

$q_3 a \to q_3 a, R$ / /прочитано $abaa$; ждем вторую букву 2-го вхождения

$q_3 b \to q_4 b, R$ / /прочитано $aba^n b, n \geq 1$

$q_4 a \to q_5 a, R$ / /найдено 2-е вхождение, поиск закончен, условие

не выполнено

$q_4 b \to q_6 b, R$ / /прочитано $aba^n bb, n \geq 1$; продолжаем искать 2-е

вхождение; состояние q_6 может быть охарактеризовано как

состояние начального поиска 2-го вхождения после прочтения

указанного выше префикса, т.е. после "разрывающей" буквы b

$q_6 a \to q_7 a, R$

$q_6 b \to q_6 b, R$

$q_7 a \to q_7 a, R$

$q_7 b \to q_8 b, R$

$q_8 a \to q_5 a, R$ / / найдено 2-е вхождение, заведомо не пересекающееся

с 1-м

$q_8 b \to q_6 b, R$

$q_5 \alpha \to q_5 \alpha, R$ / /$\alpha \in \{a, b\}$; по окончании поиска входное слово

читается до конца (т.е. до 1-го пробела)

$q_5 \square \to q_9 \square, L$ / /начало возврата головки в случае "неуспеха"

(т.е. обнаружения 2-го вхождения)

$r \square \to q_9 \square, L$ / /$r \in \{q_{00}, q_1, q_2\}$; начало возврата головки в случае "неуспеха"

(в данном случае не найдено ни одного вхождения)

$q_i \square \to q_{10} \square, L$ / /$i \in \{3, 4, 6, 7, 8\}$; начало возврата головки в случае "успеха"

(т.е. найдено ровно одно вхождение искомого слова)

$q_9 \alpha \to q_9 \square, L$

$q_{10} \alpha \to q_{10} \square, L$

$q_9 * \to q_9 *, R$

$q_9 \square \to q_f\, 0, L$

$q_{10} * \to q_{10} *, R$; $q_{10} \square \to q_f\, 1, L$

Пример работы (прогонка):

$(q_0, \lambda, *bababa\square) \mapsto (q_0, *, bababa\square) \mapsto (q_{00}, *b, ababa\square) \mapsto (q_1, *ba, baba\square) \mapsto (q_2, *bab, aba\square) \mapsto$
$(q_3, *baba, ba\square) \mapsto (q_4, *babab, a\square) \mapsto (q_5, *bababa, \square) \mapsto (q_9, *, babab, a\square) \mapsto (q_9, *, babab, \square\square) \mapsto$
$\mapsto ... \mapsto (q_9, \lambda, *\square) \mapsto (q_9, *, \square) \mapsto (q_f, \lambda, *0)$

Из состояния q4 по букве b нельзя вернуться в состояние q3, так как это может привести к ошибке:

q3→a q3→b q4 → b q3 →b q4 →a q5,

то есть вхождение abbba будет распознано как aba.
Можно уменьшить число состояний, написав программу:

$q_0 * \to q_0 *, R$

$q_0 a \to q_1 a, R$ / /буква a может быть первой буквой искомого слова;
переход в состояние q_1 означает ожидание второй буквы, т.е. буквы b

$q_0 b \to q_0 b, R$

$q_1 a \to q_1 a, R$ / /"зависаем" в состоянии q_1, ожидая букву b

$q_1 b \to q_2 b, R$ / / прочитан префикс ab

$q_2 a \to q_3 a, R$ / / найдено 1-е вхождение искомого слова, но нужно проверить наличие 2-го вхождения и при этом надо учесть, что последняя буква 1-го вхождения может оказаться первой буквой 2-го

$q_2 b \to q_0 b, R$ / /прочитан префикс abb, поиск начинается заново

$q_3 a \to q_3 a, R$ / /прочитано $abaa$; ждем вторую букву 2-го вхождения

$q_3 b \to q_4 b, R$ / /прочитано $aba^n b, n \geq 1$

$q_4 a \to q_5 a, R$ / /найдено 2-е вхождение, поиск закончен, условие не выполнено

$q_4 b \to q_6 b, R$ / /прочитано $aba^n bb, n \geq 1$; продолжаем искать 2-е вхождение; состояние q_6 может быть охарактеризовано как состояние начального поиска 2-го вхождения после прочтения указанного выше префикса, т.е. после "разрывающей" буквы b

$q_6 a \to q_3 a, R$ / /можно вернуться в состояние q_3

$q_6 b \to q_6 b, R$

$q_5 \alpha \to q_5 \alpha, R$ / / $\alpha \in \{a, b\}$; по окончании поиска входное слово читается до конца (т.е. до 1-го пробела)

$q_5 \square \to q_7 \square, L$ / /начало возврата головки в случае "неуспеха" (т.е. обнаружения 2-го вхождения)

$r \square \to q_7 \square, L$ / / $r \in \{q_0, q_1, q_2\}$; начало возврата головки в случае "неуспеха" (в данном случае не найдено ни одного вхождения, здесь же и реакция на пустое слово

$q_i \square \to q_8 \square, L$ / / $i \in \{3, 4, 6\}$; начало возврата головки в случае "успеха" (т.е. найдено ровно одно вхождение искомого слова)

$q_7 \alpha \to q_7 \square, L$

$q_8 \alpha \to q_8 \square, L$

$q_7 * \to q_7 *, R$

$q_7 \square \to q_f 0, L$

$q_8 * \to q_8 *, R;$

$q_8 \square \to q_f 1, L$

2. Сдвиг на ячейку вправо: $(q_0, \lambda, *x) \mapsto^* (q_f, \lambda, *\# x)$, где $x \in V^*$ для некоторого алфавита V, причем $\# \notin V$.

$q_0 * \to q_0 *, R$

$q_0 \square \to q_f \#, L$ / /если входное слово пустое, то в первую после маркера ячейку пишем # и заканчиваем

$q_0 \alpha \to q_\alpha \#, R$ / / $\alpha \in V$

$q_\alpha \beta \to q_\beta \alpha, R$ / / $\alpha, \beta \in V$

$q_\alpha \square \to q_1 \alpha, L$

$q_1 \alpha \to q_1 \alpha, L$

$q_1 \# \to q_f \#, L$

Тут $\alpha, \beta \in V.$

Можно реализовать сдвиг на фиксированное число k ячеек, если состояния q_α дополнительно маркировать номером итерации.

Программа:

$q_0 * \to q_0 *, R$

$q_0 \square \to q_\#^1 \#, R$ / /пустое слово на входе: пишем k решеток

$q_\#^i \# \to q_\#^{i+1} \#, R$ / / $i = \overline{1, k-1}$

$q_\#^k \# \to q_\#^R \#, L$

$q_0 \alpha \to q_\alpha^1 \#, R$ / / $\alpha \in V$

$q_\alpha^i \beta \to q_\beta^i \alpha, R$ / / $\alpha, \beta \in V, i = \overline{1, k}$

$q_\alpha^i \square \to q_1^i \alpha, L$ / / $i = \overline{1, k}$

$q_1^i \alpha \to q_1^i \alpha, L$ / / $i = \overline{1, k}$

$q_1^i \# \to q_2^i \#, R$ / / $i = \overline{1, k-1}$

$q_2^i \alpha \to q_\alpha^{i+1} \#, R$ / / $i = \overline{1, k-1}$

$q_1^k \# \to q_\#^R \#, L$

3. Сдвиг на ячейку влево: $(q_0, \lambda, *\# x) \mapsto^* (q_0, \lambda, * x)$, где $x \in V^*$ для некоторого алфавита V, причем $\# \notin V$.

$q_0 * \to q_0 *, R$

$q_0 \# \to q_\# \#, R$

$q_\# \alpha \to q_\alpha \#, L$

$q_\alpha \# \to q_0 \alpha, R$

$q_\# \square \to q_1 \square, L$

$q_1 \# \to q_1 \square, L (+)$

$q_1 \alpha \to q_1 \alpha, L$

$q_1 * \to q_f *, S$

Решетка протаскивается сквозь все буквы слова и исчезает (или исчезает сразу, если слово пустое).
Если в начальной конфигурации перед словом стоит несколько решеток, программу надо поменять так:

$q_1 \# \to q_2 \square, L$ вместо команды(+)

$q_2 \alpha \to q_2 \alpha, L$

$q_2 \# \to q_\# \#, R$

$q_\# \# \to q_\# \#, R$ (при первом проходе через #)

$q_2 * \to q_f *, S$

Другое, более эффективное, решение: читаем входное слово до конца, а потом поступаем так же при движении головки справа налево, как и в предыдущей задаче.
Запишем программу для МТ, которая из начальной конфигурации $(q_0, \lambda, *\#\overset{k}{\ldots}\# x), x \in V^*$ выводит заключительную конфигурацию $(q_0, \lambda, *x)$.

$q_0 * \to q_0 *, R$

$q_0 \alpha \to q_0 \alpha, R$ / /$\alpha \in V \cup \{\#\}$

$q_0 \square \to q_1 \square, L$

$q_1 \# \to q_1 \square, L$ / /пустое слово на входе; просто стираем все "решетки"

$q_1 * \to q_f *, S$ / /и заканчиваем

$q_1 \alpha \to q_\alpha \square, L$ / /последняя буква сдвигаемого слова стирается
и запоминается в состоянии; $\alpha \in V$

$q_\alpha \beta \to q_\beta \alpha, L$ / /команды "обмена"; $\alpha, \beta \in V$

$q_\alpha \# \to q_\# \alpha, L$ / / очередная стираемая "решетка" заменяется 1-й буквой
сдвигаемого слова; $\alpha \in V$

$q_\# \# \to q_0 \#, R$ / /если стертая "решетка" не последняя, процесс повторяется

$q_\# * \to q_f *, S$ / /стертая "решетка" последняя, процесс завершается

Пример работы (прогонка):
$(q_0, \lambda, *\#\# abc\square) \mapsto (q_0, *, \#\# abc\square) \mapsto (q_0, *\#, \# abc\square) \mapsto (q_0, *\#\#, abc\square)$
$\mapsto (q_0, *\#\# a, bc\square) \mapsto (q_0, *\#\# ab, c\square) \mapsto (q_0, *\#\# abc, \square)$
$\mapsto (q_1, *\#\# ab, c\square) \mapsto (q_c, *\#\# a, b\square\square) \mapsto (q_b, \lambda, *\#\#, ac\square)$
$\mapsto (q_a, *\#, \# bc\square) \mapsto (q_\#, *, \# abc\square) \mapsto (q_0, *\#, abc\square) \mapsto \ldots$

Процесс стирания второй «решетки» проходит аналогично.
Следует напомнить, что в рассматриваемой модели МТ лента предполагается полубесконечной (т.е. имеющей первую ячейку и не имеющей последней). Внутри входного слова «пробелы» (символы пустой ячейки) не допускаются, первый «пробел» означает конец слова (если он идет сразу за маркером начала ленты, то слово пустое), а за ним идет бесконечный «хвост» «пробелов». В записи конфигурации обычно пишется один пробел, но иногда, чтобы подчеркнуть акт стирания буквы, может быть записано несколько пробелов.

4. Построить МТ, вычисляющую следующую функцию:

$$f(x)=\begin{cases}1, \text{ если слово } aab \text{ входит в слово } x\in\{a,b,c\}^* \text{ или слово}\\ cab \text{ входит в } x\\ 0 \text{ иначе}\end{cases}$$

Программа на следующей странице.

Заметим, что эта программа ищет вхождение какого-то одного из двух слов: успех выдается, когда любое из них будет найдено первым.

Программа усложнится, если потребовать нахождение хотя бы одного вхождения каждого из двух слов. Написание такой программы может быть дано в качестве домашнего задания.

Еще задачи на написание программ вычисления характеристических функций для самостоятельного решения:

5. Поиск не менее двух вхождений слова abab
6. Поиск хотя бы одного вхождения хотя бы одного из слов aba или bab
7. Поиск хотя бы одного вхождения каждого из слов aba и bab.

$q_0 * \to q_0 *, R$

$q_0 a \to q_1^a a, R$ / /возможно, это 1-я буква 1-го слова

$q_0 b \to q_0 b, R$

$q_0 c \to q_1^c c, R$ / /возможно, это 1-я буква 2-го слова

$q_1^a a \to q_2^a a, R$ / /прочитано aa

$q_1^a b \to q_0 b, R$ / /прочитано ab, возврат к началу поиска

$q_1^a c \to q_1^c c, R$ / /прочитано ac; буква c может быть 1-й буквой 2-го слова

$q_2^a a \to q_2^a a, R$ / /ждем 3-ю букву 1-го слова

$q_2^a b \to q_3 b, R$ / / найдено 1-е слово

$q_2^a c \to q_1^c c, R$ / / прочитано aac

$q_1^c a \to q_2^c a, R$ / /прочитано ca

$q_1^c b \to q_0 b, R$ / /прочитано cb, возврат к началу поиска

$q_1^c c \to q_1^c c, R$ / /ждем 2-ю букву 2-го слова

$q_2^c a \to q_2^a a, R$ / /прочитано caa (если следующая буква окажется b, то будет найдено 1-е слово)

$q_2^c b \to q_3 b, R$ / /найдено 2-е слово

$q_2^c c \to q_1^c c, R$ / /прочитано cac

$q_3 \alpha \to q_3 \alpha, R$ / / $\alpha \in \{a, b, c\}$; дочитывание входного слова по окончании поиска

$q_3 \square \to q_4 \square, L$ / /начало возврата при успешном поиске

$q_4 \alpha \to q_4 \square, L$

$q_4 * \to q_4 *, R$

$q_4 \square \to q_f 1, L$ / / успешное окончание работы(по крайней мере, одно из слов найдено)

$r \square \to q_5 \square$/ / $r \in \{q_0, q_1^a, q_2^a, q_1^c, q_2^c\}$; начало возврата при неуспешном поиске

$q_5 \alpha \to q_5 \square, L$

$q_5 * \to q_5 *, R$

$q_5 \square \to q_f 0, L$ / / неуспешное окончание работы(ни одно слово не найдено)

8. Написать программу МТ, удваивающую произвольное слово в некотором произвольно заданном алфавите.

Нужно написать программу МТ, которая из начальной конфигурации $(q_0, \lambda, *x\square)$ выводит заключительную конфигурацию $(q_f, \lambda, *xx\square)$, где $x \in V*$ для произвольно заданного алфавита $V = \{a_1, ..., a_n\}$.

Для решения введем алфавит букв-«двойников» $\overline{V} = \{\overline{a}_1, ..., \overline{a}_n\}$, находящийся во взаимно однозначном соответствии с исходным и не пересекающегося с ним. Пусть также символ # не входит в объединенный алфавит $V \cup \overline{V}$.

Программа:

$q_0 * \to q_o *, R$

$q_0 \square \to q_f \square, L$ / / пустое входное слово

$q_0 \alpha \to q_\alpha \#, R$ / / $\alpha \in V$; первая (а далее очередная копируемая) буква входного слова "забивается" решеткой и запоминается в состоянии, а решетка отмечает левую границу копирования

$q_\alpha \beta \to q_\alpha \beta, R$ / / $\alpha, \beta \in V$; копируемая буква проносится через все буквы исходного слова и через частично построенную копию (следующая строка)

$q_\alpha \overline{\beta} \to q_\alpha \overline{\beta}, R$ / / $\alpha, \beta \in V$

$q_\alpha \square \to q_\alpha^{rev} \overline{\alpha}, L$ / / в первую пустую ячейку записывается очередная копия и головка движется назад за следующей буквой

$q_\alpha^{rev} \overline{\beta} \to q_\alpha^{rev} \overline{\beta}, L$ / / $\alpha, \beta \in V$

$q_\alpha^{rev} \beta \to q_\alpha^{rev} \beta, L$ / / $\alpha, \beta \in V$

$q_\alpha^{rev} \# \to q_0 \alpha, R$ / / скопированная буква восстанавливается и головка обозревает либо следующую букву, которую надо скопировать, либо копию первой буквы (последнее означает, что копия входного слова построена и записана в алфавите "двойников")

$q_0 \overline{\alpha} \to q_1 \alpha, R$ / / $\alpha \in V$; копия входного слова переводится в алфавит V

$q_1 \overline{\alpha} \to q_1 \alpha, R$

$q_1 \square \to q_2 \square, L$

$q_2 \alpha \to q_2 \alpha, L$ / / $q_2 -$ состояние окончательного возврата после завершения копирования ($\alpha \in V$)

$q_2 * \to q_f *, S$

Прогонка для слова abc:

$(q_0, \lambda, *abc\square) \mapsto (q_0, *, abc\square) \mapsto (q_a, *\#, bc\square) \mapsto (q_a, *\#b, c\square)$

$\mapsto (q_a, *\#bc, \square) \mapsto (q_a^{rev}, *\#b, c\overline{a}\square) \mapsto (q_a^{rev}, *\#, bc\overline{a}\square) \mapsto (q_a^{rev}, *, \#bc\overline{a}\square)$

$\mapsto (q_0, *a, bc\overline{a}\square) \mapsto (q_b, *a\#, c\overline{a}\square) \mapsto^2 (q_b, *a\#c\overline{a}, \square) \mapsto (q_b^{rev}, *a\#c\overline{a}, \overline{b}\square)$

$\mapsto^3 (q_b^{rev}, *a, \#c\overline{a}\overline{b}\square) \mapsto (q_0, *ab, c\overline{a}\overline{b}\square) \mapsto (q_c, *ab\#, \overline{a}\overline{b}\square) \mapsto^2 (q_c, *ab\#\overline{a}\overline{b}, \square)$

$\mapsto (q_c^{rev}, *ab\#\overline{a}\overline{b}, \overline{c}\square) \mapsto^3 (q_c^{rev}, *ab, \#\overline{a}\overline{b}\overline{c}\square) \mapsto (q_0, *abc, \overline{a}\overline{b}\overline{c}\square) \mapsto (q_1, *abca, \overline{b}\overline{c}\square)$

$\mapsto^2 (q_1, *abcabc, \square) \mapsto (q_2, *abcab, c\square) \mapsto^7 (q_2, \lambda, *abcabc\square) \mapsto (q_f, \lambda, *abcabc\square)$

Число над значком $\mapsto$ (непосредственной выводимости) означает число шагов между соседними конфигурациями (в том случае, если оно больше 1).

В качестве задачи для самостоятельного решения можно предложить написать программу МТ, инвертирующую исходное слово.

9 (задача 1-го варианта ДЗ). Пусть V - произвольный алфавит, $u = u(1)u(2)...u(k), k \geq 1$ - произвольно фиксированное непустое слово в этом алфавите.
Написать программу МТ, которая аннулирует (т.е. перерабатывает в пустое слово) те и только те слова в алфавите V, которые содержат вхождение слова u.
Иначе говоря, МТ должна вычислять такую функцию:

$$f(x) = \begin{cases} \lambda, & \text{если } u \text{ входит в } x \\ x, & \text{если } u \text{ не входит в } x \text{ и } x \neq \lambda \\ \#, & \text{если } x = \lambda (\# \notin V) \end{cases}$$

То есть МТ должна стирать «правильное» слово, оставлять без изменений «неправильное» непустое слово и выдавать некий «спецсимвол» (типа сигнала ошибки), если на входе пустое слово. Пустое слово, очевидно, не является «правильным», но оставлять его без изменений было бы некорректно с учетом того, что пустое слово как результат выдается именно для «правильного» слова.
Некоторая трудность задачи состоит в том, что программа МТ должна быть записана в общем виде для **произвольного** слова u, наличие вхождения которого во входное слово проверяется. С содержательной точки зрения это означает, что некоторый внешний эмулятор программы слово u получает как параметр.
Идея работы программы достаточно проста. Машина ищет первое вхождение первой буквы слова и, найдя его, путем смены состояний фиксирует следующие буквы слова. Но нужно учесть, что слово может оборваться, т. е. будет найден некоторый его начальный фрагмент (который, в частности, может пересекаться с самим словом:
x = *ab**ababaab***, а u = *ababaab* – здесь обрыв произойдет на 6-й букве). Тогда головка должна вернуться и начать новый поиск (следующего вхождения 1-й буквы слова u). Но чтобы программа не зациклилась, возврат не должен идти левее ранее фиксированного вхождения 1-й буквы, для чего её следует специально пометить, используя, например, прием введения алфавита букв-«двойников» (см. предыдущую задачу).
Следовательно, рабочий алфавит МТ должен содержать исходный алфавит V, алфавит «двойников» и «решетку».
Программа может быть записана так (следующая страница):

$q_0 * \to q_0 *, R$

$q_0 \square \to q_f \#, L / / x = \lambda$

$q_0 \alpha \to q'_0 \alpha, R / / \alpha \in V, \alpha \neq u(1)$

$q'_0 \alpha \to q'_0 \alpha, R / / \alpha \in V, \alpha \neq u(1)$

$q_0 u(1) \to q_1 \bar{u}(1), R$

$q'_0 u(1) \to q_1 \bar{u}(1), R$ / /1-я буква искомого вхождения заменяется своим "двойником"

$q_i u(i+1) \to q_{i+1} u(i+1), R / / i = 1, \ldots, k-1$

$q_k \alpha \to q_k \alpha, R / /$ вхождение найдено, входное слово дочитывается до конца; $\alpha \in V$

$q_k \square \to q_{rev} \square, L / /$ *возврат* (при успешном поиске)

$q_{rev} \alpha \to q_{rev} \square, L / / \alpha \in V \cup \bar{V}$

$q_{rev} * \to q_f *, S$

$q'_0 \square \to r \square, L / /$ входное слово (непустое) прочитано, искомое вхождение не найдено; r - состояние возврата при неуспешном поиске (заметим, что нельзя было при поиске в непустом слове оставить состояние q_0, так как это привело бы к конфликту команд: возникли бы две команды с левой частью $q_0 \square$)

$q_i \square \to r \square, L / / i = \overline{1, k-1;}$ еще одна ситуация неуспеха: слово обрывается, и головка оказывается сразу на пустой ячейке (это означает, что искомого вхождения заведомо нет)

$r \alpha \to r \alpha, L / / \alpha \in V$

$r \bar{\alpha} \to r \alpha, L / / \alpha \in V;$ все "двойники" заменяются оригиналами, входное (непустое) слово восстанавливается

$r * \to q_f *, S$

$q_i \beta \to \bar{q} \beta, L / / i = 1, \ldots, k-1; \beta \neq u(i+1), \beta \neq \square$. Обрыв искомого вхождения; состояние $\bar{q}$ есть состояние "промежуточного" возврата

$\bar{q} \alpha \to \bar{q} \alpha, L / / \alpha \in V$

$\bar{q} \bar{u}(1) \to q'_0 u(1), R / /$ начало новой итерации поиска после обрыва искомого вхождения.