УДК 519

П. В. Мартюгин

PSPACE-ПОЛНОТА ЗАДАЧИ ПРОВЕРКИ ЧАСТИЧНЫХ АВТОМАТОВ НА БЕРЕЖНУЮ СИНХРОНИЗИРУЕМОСТЬ*

1. Введение

Частичным конечным автоматом (*ЧКА*) называется тройка $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, Σ – конечный *алфавит* и δ – частичная *функция переходов*, действующая из $Q \times \Sigma$ в Q (последнее означает, что функция δ может быть неопределена на некоторых парах из множества $Q \times \Sigma$). Обозначим через 2^Q множество всех подмножеств множества Q, а через Σ^* – свободный моноид, порожденный алфавитом Σ с пустым словом λ. Функция δ может быть естественным образом продолжена на множество $2^Q \times \Sigma^*$. Положим для всех $q \in Q$, $\delta(q, \lambda) = q$. Пусть $q \in Q$, $w = ua, a \in \Sigma$, $u \in \Sigma^*$. Если определено состояние $p = \delta(q, u)$ и определено состояние $\delta(p, a)$, то положим $\delta(q, w) = \delta(p, a)$. Если $S \subseteq Q$, $w \in \Sigma^*$ и функция $\delta(q, w)$ определена на всех состояниях $q \in S$. Тогда положим $\delta(S, w) = \{\delta(q, w) \mid q \in S\}$. Часто нам будет удобнее использовать и другое обозначение $S.w$ для $\delta(S, w)$.

ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ называется *бережно синхронизируемым*, если существует слово $w \in \Sigma^*$ такое, что величина $\delta(Q, w)$ определена и $|\delta(Q, w)| = 1$. Такое слово w мы будем называть *бережно синхронизирующим* словом (*б.с.с.*) для автомата $\mathscr{A}$. Очевидно, что для каждого бережно синхронизируемого автомата найдется кратчайшее бережно синхронизирующее слово.

Синхронизацию удобно себе представлять как передвижение фишек, стоящих на состояниях автомата. Пусть рассматривается бережно синхронизируемый ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$ и w – б.с.с. для $\mathscr{A}$. Представим себе, что в начальный момент времени на всех состояниях автомата стоит по *фишке*. Пусть $a \in \Sigma$ и на состоянии $q \in Q$ стоит фишка, тогда под действием буквы a эта фишка *передвигается* на состояние $\delta(q, a)$. Если несколько фишек оказываются на одном состоянии, то они *склеиваются* в одну. Если буква $a \in \Sigma$ не определена на состоянии $q \in Q$ и на состоянии q стоит фишка, то после действия буквы a фишка *умирает*. Б.с.с. $w \in \Sigma^*$ переводит фишки со всех состояний автомата на одно таким образом, что ни одна из них не умирает. Поэтому синхронизация и называется *бережной*.

*Работа выполнена при поддержке программы «Развитие научного потенциала высшей школы», проект № 2.1.1/3537.

Бережная синхронизируемость ЧКА является естественным обобщением понятия синхронизируемости детерминированных конечных автоматов (ДКА) со всюду определенной функцией переходов. Пусть $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ – ДКА, слово $w \in \Sigma^*$ называется синхронизирующим для автомата $\mathscr{A}$, если $|\delta(Q, w)| = 1$. В 1964 г. Черни высказал гипотезу, что любой синхронизируемый ДКА с n состояниями может быть синхронизирован словом длины не большей $(n-1)^2$ (см. [1]). Предпринималось множество попыток доказать или опровергнуть эту гипотезу, но ни одна из них не увенчалась успехом. Гипотеза доказана только для некоторых частных случаев (см. [2–5]). В общем случае получена только кубическая верхняя оценка $(n^3 - n)/6$ (см. [6]). Соответствующая нижняя оценка максимальной длины кратчайшего синхронизирующего слова получена Черни в [1].

Аналогично можно сформулировать задачу об оценке длины кратчайшего б.с.с. для ЧКА. Обозначим через $\omega(n)$ максимально возможную длину кратчайшего б.с.с. для ЧКА с n состояниями. Первыми задачу об оценке величины $\omega(n)$ рассматривали М. Ито и К. Шикишима-Тшуджи (см. [7,8]) (в их работах эта величина обозначалась через $d_3(n)$). Они доказали, что

$$\omega(n) \leqslant 2^n - 2^{n-2} - 1, \qquad \begin{array}{l} \text{если } n = 2k, \text{ то } \omega(n) \geqslant 2^{n/2} + 1, \\ \text{если } n = 2k+1, \text{ то } \omega(n) \geqslant 3 \cdot 2^{(n-3)/2} + 1. \end{array}$$

Понятие бережной синхронизируемости было введено автором данной статьи в [9]. Там же была улучшена нижняя оценка величины $\omega(n)$. Было доказано, что

$$\begin{array}{l} \text{если } n = 3k, \text{ то } \omega(n) \geqslant 3 \cdot 3^{\frac{n}{3}} - 2; \\ \text{если } n = 3k+1, \text{ то } \omega(n) \geqslant 4 \cdot 3^{\frac{n-1}{3}} - 2; \\ \text{если } n = 3k+2, \text{ то } \omega(n) \geqslant 6 \cdot 3^{\frac{n-2}{3}} - 2; \\ \text{если } n = 3^r, \text{ то } \omega(n) \geqslant 5 \cdot 3^{\frac{n}{3}} - o(3^{\frac{n}{3}}). \end{array}$$

Первый естественный вопрос при обсуждении данного понятия состоит в том, как для заданного ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ найти бережно синхронизирующее слово. Проще всего это сделать, используя *автомат на подмножествах* $2^{\mathscr{A}} = (2^Q, \Sigma, \delta)$, где 2^Q – множество всех подмножеств множества Q, а $\delta : 2^Q \times \Sigma \to 2^Q$ – продолжение функции δ. Если существует путь в автомате $2^{\mathscr{A}}$ из множества Q в некоторое одноэлементное множество, то слово, читаемое вдоль этого пути, будет бережно синхронизирующим. Путь можно найти поиском в ширину по графу автомата $2^{\mathscr{A}}$. В этом случае найденное слово будет кратчайшим. Описанный алгоритм требует экспоненциального времени и экспоненциальной памяти.

Какова же сложность задачи вычисления бережно синхронизирующего слова для заданного ЧКА? Задача поиска слова, синхронизирующего данный ДКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, решается за время $\Theta(|\Sigma| \cdot |Q|^2 + |Q|^3)$ (см. [3]), т. е. за полиномиальное время. Задача поиска кратчайшего синхронизирующего слова для данного ДКА NP-трудна и co-NP-трудна одновременно (см. [10]). В этой статье мы докажем, что задача проверки заданного ЧКА на бережную синхронизируемость PSPACE-полна даже в случае автоматов с двумя буквами. Также мы докажем, что PSPACE-полны: задача проверки заданного автомата на бережную синхронизируемость словом заданной длины и задача поиска длины кратчайшего б.с.с. для заданного ЧКА.

Введем несколько дополнительных обозначений. Пусть w – слово над алфавитом Σ. Обозначим через $|w|$ длину слова w. Для $i \in 1 \dots |w|$ через $w[i]$ обозначим i-ю букву слова w. Для $i, j \in 1 \dots |w|$ через $w[i, j]$ обозначим слово $w[i]w[i+1] \dots w[j]$.

2. Рассматриваемые задачи

Сформулируем основные задачи, сложность которых мы будем обсуждать.

Задача: БЕРЕЖНАЯ СИНХРОНИЗИРУЕМОСТЬ (БЕР СИНХ)
Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$.
Вопрос: Существует ли б.с.с. $w \in \Sigma^*$ для автомата $\mathscr{A}$?

Задача: k-БЕРЕЖНАЯ СИНХРОНИЗИРУЕМОСТЬ (k-БЕР СИНХ)
Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, $|\Sigma| = k$.
Вопрос: Существует ли б.с.с. $w \in \Sigma^*$ для автомата $\mathscr{A}$?

Аналогичные задачи для ДКА имеют решение за полиномиальное время. Существует алгоритм, который проверяет заданный ДКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ на синхронизируемость и работает за время $\Theta(|\Sigma| \cdot |Q|^2)$ (см. [3]).

Задача: БЕРЕЖНО СИНХРОНИЗИРУЮЩЕЕ СЛОВО ДЛИНЫ L (БЕР СИНХ ДЛИНЫ L)
Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$ и целое число $L > 0$.
Вопрос: Существует ли б.с.с. для автомата $\mathscr{A}$ длины, не большей чем L?

Задача: k-БЕРЕЖНО СИНХРОНИЗИРУЮЩЕЕ СЛОВО ДЛИНЫ L (k-БЕР СИНХ ДЛИНЫ L)
Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, $|\Sigma| = k$ и целое число $L > 0$.

Вопрос: Существует ли б.с.с. для автомата $\mathscr{A}$ длины, не большей чем L?

Аналогичные задачи для ДКА (для заданного автомата $\mathscr{A}$ и числа L указать, существует ли слово длины $\leqslant L$, синхронизирующее автомат $\mathscr{A}$) являются NP-полными (см. [11]).

Задача: КРАТЧАЙШЕЕ БЕРЕЖНО СИНХРОНИЗИРУЮЩЕЕ СЛОВО (КРАТ БЕР СИНХ)

Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$и целое число $L > 0$.

Вопрос: Верно ли, что кратчайшее б.с.с. для автомата $\mathscr{A}$ имеет длину L?

Задача: КРАТЧАЙШЕЕ k-БЕРЕЖНО СИНХРОНИЗИРУЮЩЕЕ СЛОВО (КРАТ k-БЕР СИНХ)

Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, $|\Sigma| = k$ и целое число $L > 0$.

Вопрос: Верно ли, что кратчайшее б.с.с. для автомата $\mathscr{A}$ имеет длину L?

Аналогичные задачи для ДКА (для заданного автомата $\mathscr{A}$ и числа L указать, имеет ли кратчайшее слово, синхронизирующее автомат $\mathscr{A}$, длину L) являются одновременно NP-трудными и co-NP-трудными, однако скорее всего не лежат ни в NP, ни в co-NP (см. [10]).

Задача: 3-ВЫПОЛНИМОСТЬ С КВАНТОРАМИ (3-ВЫП С КВАНТОРАМИ)

Дано: Формула $F = Q_1x_1Q_2x_2 \dots Q_nx_nP(x_1 \dots x_n)$, где $n > 1$ и $x_1, \dots, x_n$ – булевы переменные; $Q_1, \dots, Q_n \in \{\exists, \forall\}$ – кванторы существования или всеобщности; $P(x_1 \dots x_n)$ – булева формула в конъюнктивной нормальной форме вида $\bigwedge\limits_{\ell=1}^{N}(y_1^\ell \vee y_2^\ell \vee y_3^\ell)$, где $y_1^\ell, y_2^\ell, y_3^\ell \in \{x_i | i \in 1 \dots n\} \cup \{\neg x_i | i \in 1 \dots n\}$ для $\ell \in \{1, \dots, N\}$, $N \geqslant 1$.

Вопрос: Является ли формула F истинной?

Задача 3-ВЫП С КВАНТОРАМИ – это классическая PSPACE-полная задача (см. [12]).

Пусть задача A полиномиально сводима к задаче B. В этом случае мы будем писать $A \leqslant_p B$.

Предложение 1. *Для любого $k \geqslant 2$*

1. k-БЕР СИНХ $\leqslant_p$ $(k+1)$-БЕР СИНХ
2. k-БЕР СИНХ $\leqslant_p$ БЕР СИНХ

3. k-БЕР СИНХ ДЛИНЫ $L \leqslant_p (k+1)$-БЕР СИНХ ДЛИНЫ L

4. k-БЕР СИНХ ДЛИНЫ $L \leqslant_p$ БЕР СИНХ ДЛИНЫ L

5. КРАТ k-БЕР СИНХ $\leqslant_p$ КРАТ $(k+1)$-БЕР СИНХ

6. КРАТ k-БЕР СИНХ $\leqslant_p$ КРАТ БЕР СИНХ

Доказательство. 1. Пусть на вход задачи k-БЕР СИНХ подается автомат $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, $|\Sigma| = k$. Построим автомат $\mathscr{A}' = \langle Q, \Sigma \cup \{a\}, \delta'\rangle$ такой, что для всех состояний $q \in Q$, $\delta'(q,a) = q$ и для всех $b \in \Sigma$, $\delta'(q,b) = \delta(q,b)$. Для любого б.с.с. $w \in \Sigma^*$ для автомата $\mathscr{A}$ оно также является б.с.с. для автомата $\mathscr{A}'$. Пусть $w \in (\Sigma \cup \{a\})^*$ – б.с.с. для автомата $\mathscr{A}'$. Рассмотрим слово $\pi(w) \in \Sigma^*$, получаемое из слова w выбрасыванием всех букв a. Заметим, что $\delta'(Q,w) = \delta'(Q,\pi(w)) = \delta(Q,\pi(w))$, поэтому $\pi(w)$ – б.с.с. для автоматов $\mathscr{A}'$ и $\mathscr{A}$. Поэтому автомат $\mathscr{A}$ бережно синхронизируем тогда и только тогда, когда автомат $\mathscr{A}'$ бережно синхронизируем. Таким образом, k-БЕР СИНХ $\leqslant_p$ $\leqslant_p (k+1)$-БЕР СИНХ.

3. Пусть на вход задачи k-БЕР СИНХ ДЛИНЫ L подается автомат $\mathscr{A} = \langle Q, \Sigma, \delta\rangle$, $|\Sigma| = k$ и число $L > 0$. Построим автомат $\mathscr{A}'$ так же, как в предыдущем случае, а число возьмем равным L. Если $w \in \Sigma^*$ – б.с.с. для автомата $\mathscr{A}$ и $|w| \leqslant L$, то w – б.с.с. для автомата $\mathscr{A}'$. Если $w \in (\Sigma \cup \{a\})^*$ – б.с.с. для автомата $\mathscr{A}'$ и $|w| \leqslant L$, то $\pi(w)$ – б.с.с. для автомата $\mathscr{A}$ и $|\pi(w)| \leqslant |w| \leqslant L$. Таким образом, k-БЕР СИНХ ДЛИНЫ $L \leqslant_p (k+1)$-БЕР СИНХ ДЛИНЫ L.

5. Автомат $\mathscr{A}'$ и соответствующее число L строим так же, как в предыдущем случае. Пусть слово $w \in \Sigma^*$ – кратчайшее б.с.с. для автомата $\mathscr{A}$ и $|w| = L$. Слово w является б.с.с. для автомата $\mathscr{A}'$. Предположим от противного, что существует слово $u \in (\Sigma \cup \{a\})^*$ – б.с.с. для автомата $\mathscr{A}'$ и $|u| < |w|$. Рассмотрим слово $\pi(u)$. Так как $\pi(u) \in \Sigma^*$, то справедливо равенство $|\delta(Q,\pi(u))| = |\delta'(Q,\pi(u))| = |\delta'(Q,u)| = 1$, поэтому $\pi(u)$ – б.с.с. для автомата $\mathscr{A}$. Но в то же время $|\pi(u)| < |u| < |w|$, что противоречит тому, что слово w – кратчайшее б.с.с. для $\mathscr{A}$. Следовательно, кратчайшее б.с.с. для автомата $\mathscr{A}'$ имеет длину L.

Пусть слово $w \in (\Sigma \cup \{a\})^*$ кратчайшее б.с.с. для автомата $\mathscr{A}'$ и $|w| = L$. Из того что $|\delta'(Q,\pi(w))| = |\delta'(Q,w)| = 1$, получаем $\pi(w) = w$. Следовательно, $w \in \Sigma^*$ и поэтому w является б.с.с. для автомата $\mathscr{A}$. Если u – б.с.с. для автомата $\mathscr{A}$ и $|u| < |w|$, то u также является б.с.с. для автомата $\mathscr{A}'$ и слово w не будет кратчайшим б.с.с. для $\mathscr{A}'$. Поэтому кратчайшее б.с.с. для автомата $\mathscr{A}$ имеет длину L. Таким образом, КРАТ k-БЕР СИНХ $\leqslant_p$ КРАТ $(k+1)$-БЕР СИНХ.

Утверждения 2, 4, 6 получаются очевидным образом, так как автоматы с k буквами являются частными случаями произвольных автоматов.

Приступаем к доказательству PSPACE-полноты всех вышеперечисленных задач, рассмотренных для любых k. Для этого мы найдем алгоритм, позволяющий решить задачи БЕР СИНХ, БЕР СИНХ ДЛИНЫ L и КРАТ БЕР СИНХ, используя дополнительную память полиномиального размера (предложение 2). Кроме того, мы полиномиально сведем задачу 3-ВЫП С КВАНТОРАМИ к задачам 2-БЕР СИНХ, 2-БЕР СИНХ ДЛИНЫ L и КРАТ 2-БЕР СИНХ (предложение 3). Таким образом, благодаря предложению 1 получим PSPACE-полноту задач БЕР СИНХ, БЕР СИНХ ДЛИНЫ L, КРАТ БЕР СИНХ, k-БЕР СИНХ, k-БЕР СИНХ ДЛИНЫ L и КРАТ k-БЕР СИНХ для всех k.

3. Принадлежность к PSPACE

В силу того что модель вычислений, использующая память произвольного доступа (RAM) полиномиально эквивалентна машине Тьюринга (см. [13]), мы будем использовать для вычислений обычный компьютер. Построим алгоритм проверки автомата $\mathscr{A}$ на бережную синхронизируемость, использующий $n^2 + O(n)$ дополнительной памяти. Идея взята из алгоритма поиска пути между двумя вершинами в заданном графе, который использует дополнительную память логарифмического размера.

Основную идею алгоритма проверки ЧКА на бережную синхронизируемость проще всего понять на примере решения более простой задачи.

Задача: ПУТЬ МЕЖДУ ДВУМЯ МНОЖЕСТВАМИ
Дано: ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, множества $T_0, T_1 \subseteq Q$ и целое число $k > 0$.
Вопрос: Верно ли, что существует слово $w \in \Sigma^*$ длины, не большей чем 2^k, такое что $T_0.w = T_1$.

Определим функцию $Path(X, Y, d)$, принимающую значения 0 и 1. Положим $Path(X, Y, d) = 1$ тогда и только тогда, когда существует слово $u \in \Sigma^*$ такое, что $X.u = Y$ и длина u не больше d. Мы будем использовать следующее очевидное свойство: пусть d – четное число, тогда $Path(X, Y, d) = 1$ в том и только в том случае, когда существует множество Z такое, что $Path(X, Z, d/2) = 1$ и $Path(Z, Y, d/2) = 1$ (т. е. Z – середина пути от X до Y).

Задача ПУТЬ МЕЖДУ ДВУМЯ МНОЖЕСТВАМИ может быть решена при помощи рекурсивного вызова функции $Path$. Сначала вызывается $Path(T_0, T_1, 2^k)$. При каждом вызове $Path(X, Y, d)$ для некоторых $X, Y \subseteq Q$ и $d > 1$ перебираются всевозможные $Z \subseteq Q$ и для каждого вызываются функции $Path(X, Z, d/2)$ и $Path(Z, Y, d/2)$. Если $Path(X, Z, d/2) = 1$ и $Path(Z, Y, d/2) = 1$, то $Path(X, Y, d) = 1$. Если $d = 1$, то просто нужно про-

верить, найдется ли буква $a \in \Sigma$ такая, что $X.a = Y$. Несложно понять, что $Path(T_0, T_1, 2^k) = 1$ тогда и только тогда, когда существует слово $w \in \Sigma^*$ такое, что $|w| \leqslant 2^k$ и $T_0.w = T_1$.

Задача БЕР СИНХ ДЛИНЫ L отличается от задачи ПУТЬ МЕЖДУ ДВУМЯ МНОЖЕСТВАМИ тем, что начальное множество T_0 всегда является всем множеством состояний Q, конечное множество T_1 может быть любым из одноэлементных подмножеств множества Q и длина слова w ограничена числом L, которое может не быть степенью двойки. Поэтому для решения задачи БЕР СИНХ ДЛИНЫ L мы опишем чуть более сложный алгоритм.

Предложение 2. *Существует алгоритм, позволяющий решить задачи* БЕР СИНХ, БЕР СИНХ ДЛИНЫ L *и* КРАТ БЕР СИНХ, *используя дополнительную память полиномиального размера.*

Доказательство. Построим функцию, которая реализует алгоритм, решающий задачу БЕР СИНХ ДЛИНЫ L. Алгоритмы решения задач БЕР СИНХ и КРАТ БЕР СИНХ будут использовать эту функцию в качестве подпрограммы. Пусть на вход нам даны ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, $Q = \{1, \ldots, n\}$, $\Sigma = \{a_1, \ldots, a_m\}$ и целое число $L > 0$. Построим алгоритм, отвечающий на вопрос: существует ли слово $w \in \Sigma^*$ длины, не большей чем L, бережно синхронизирующее автомат $\mathscr{A}$.

Рассмотрим автомат $2^{\mathscr{A}} = (2^Q, \Sigma, \delta)$, где 2^Q – множество всех подмножеств множества Q, а $\delta : 2^Q \times \Sigma \to 2^Q$ – продолжение функции δ. Если $w \in \Sigma^*$ – б.с.с. для автомата $\mathscr{A}$, то в автомате $2^{\mathscr{A}}$ слово w переводит множество Q в некоторое одноэлементное множество. Пусть слово w кратчайшее б.с.с. для автомата $\mathscr{A}$ и $|w| = t$, тогда существуют множества $S[0], S[1], \ldots, S[t] \subseteq Q$ такие, что $S[0] = Q, |S[t]| = 1$ и $S[i-1].w[i] = S[i]$ для всех $i = 1, \ldots, t$. Если для некоторых $i \neq j$, $S[i] = S[j]$, то слово $w[1, i]w[j+1, t]$ также является б.с.с. для $\mathscr{A}$. В этом случае слово w не является кратчайшим б.с.с. Поэтому можно считать, что $S[i] \neq S[j]$ для $i \neq j$. Длина кратчайшего б.с.с. для автомата с n состояниями не больше чем $2^n - 2^{n-2} - 1$ [см]. Поэтому $t \leqslant 2^n - 2^{n-1} - 1 < 2^n - 2$.

Каждое множество $S[i]$ может быть закодировано при помощи n бит

$$S_1[i], \ldots, S_n[i], \quad S_\eta[i] = \begin{cases} 0, & \eta \notin S[i], \\ 1, & \eta \in S[i]. \end{cases}$$

Введем массив $B[j]$, $j = 0 \ldots n$, элементами которого будут n-битные записи множеств из $\{S[i] | i = 1 \ldots n\}$. Элемент $B[j]$ служит для хранения подмножеств множества Q, претендующих на то, чтобы быть множествами $S[\mu \cdot 2^j]$ для различных нечетных μ.

В начале работы алгоритма положим $B[n] = Q$. Основой алгоритма будет служить рекурсивно вызываемая функция $InPath(k, c, L)$. При вызове функция $InPath(k, c, L)$ перебирает различные значения элемента $B[k]$ и проверяет, может ли $B[k]$ быть равным $S[c]$. Значение $InPath(k, c, L) = Bad$, если на данном шаге очередное множество $S[c]$ не найдено или $c > L$; $InPath(k, c, L) = Good$, если на данном шаге очередное множество $S[c]$ найдено и $c \leqslant L$, но $|S[c]| \neq 1$; $InPath(k, c, L) = Found$, если множество $S[c]$ найдено, $c \leqslant L$ и $|S[c]| = 1$. Массив $B[j]$ является глобальным. Проверка существования б.с.с. длины, не большей чем L, для автомата $\mathscr{A}$ осуществляется запуском функции $InPath(n-1, 2^{n-1}, L)$. Автомат $\mathscr{A}$ является глобальной переменной. Функция $InPath$ выглядит следующим образом:

Функция $InPath(k, c, L)$
Если $k > 0$, **то**
    $e = Bad$
    Пока $e = Bad$ **перебирать все** множества $T \subseteq Q$
        $B[k] = T$
        $e = InPath(k-1, c - 2^{k-1}, L)$
        Если $e = Good$, **то**
            Если $|T| = 1$ **и** $c \leqslant L$, **то**
                $e = Found$
            Иначе
                $e = InPath(k-1, c + 2^{k-1}, L)$
    Вернуть $InPath = e$

Если $k = 0$, **то**
    $\ell =$ Номер младшего ненулевого разряда
    в двоичной записи числа $c + 1$
    Если $c = 1$, **то**
        $r = n$
    Иначе
        $r =$ Номер младшего ненулевого разряда
        в двоичной записи числа $c - 1$

    Искать множество $T \subseteq Q$ **такое, что**
        найдется буква $a \in \Sigma$ **такая, что** $B[r].a = T$
    и
        ($|T| = 1$ **и** $c \leqslant L$) **или найдется** буква $b \in \Sigma$
        такая, что $T.b = B[\ell]$
    Если множество T нашлось, **то**

$B[0] = T$
Если $|T| = 1$, **то**
Вернуть $InPath = Found$
Иначе
Вернуть $InPath = Good$
Иначе
Вернуть $InPath = Bad$
Конец функции

Алгоритм, решающий задачу БЕР СИНХ ДЛИНЫ L, выглядит следующим образом:

Функция $CheckLength(\mathscr{A}, L)$
$B[n] = Q$
Если $InPath(n-1, 2^{n-1}, L) = Found$, **то**
Вернуть Для автомата $\mathscr{A}$ существует б.с.с. длины L
Иначе
Вернуть Для автомата $\mathscr{A}$ не существует б.с.с. длины L
Конец функции

Алгоритм работает следующим образом. Производится поиск пути в автомате $2^{\mathscr{A}}$ из множества Q в одноэлементное множество. В элементе $B[n]$ всегда хранится множество $S[0] = Q$. При запуске $InPath(n-1, 2^{n-1}, L)$ в элемент $B[n-1]$ поочередно кладутся все множества $T \subseteq Q$ и производится проверка, может ли множество $S[2^{n-1}]$ быть равным $B[n-1]$. Для этого рекурсивно вызываются $InPath(n-2, 2^{n-2}, L)$ и $InPath(n-2, 3 \cdot 2^{n-2}, L)$. В первый раз производится проверка, может ли множество $S[2^{n-2}]$ быть равным $B[n-2]$ для всех возможных $B[n-2] = T \subseteq Q$. Во второй раз производится проверка, может ли множество $S[3 \cdot 2^{n-2}]$ быть равным $B[n-2]$ для всех возможных $B[n-2] = T \subseteq Q$. При каждой из этих проверок, в свою очередь, по два раза вызывается $InPath(n-3, c, L)$ для разных c и т. д.

Рассмотрим пример работы алгоритма на автомате Черни с тремя состояниями

$$\mathscr{A} = (\{1,2,3\}, \{a_1, a_2\}, \delta),$$

$$1.a_1 = 1, \quad 2.a_1 = 2, \quad 1.a_2 = 2, \quad 2.a_2 = 3, \quad 3.a_1 = 3.a_2 = 1.$$

Пусть $L = 4$.

В начале работы алгоритма присваивается $B[3] = \{1,2,3\}$. Вызывается $InPath(2,4,4)$. Перебираются различные варианты множества $B[2]$. Пусть $B[2] = \{1\}$, это будет $S[2^{n-1}] = S[4]$.

Затем вызывается $InPath(1, 2, 4)$. Перебираются различные варианты множества $B[1]$. Пусть $B[1] = \{2, 3\}$, это будет $S[2^{n-2}] = S[2]$.

Затем вызывается $InPath(0, 1, 4)$. Перебираются различные варианты множества $B[0]$. Пусть $B[0] = \{1, 2\}$, это будет $S[2^{n-3}] = S[1]$. Вычисляются $r = 0, \ell = 2$. Находятся буквы $a = a_1$, $b = a_2$ такие, что

$$S[0].a = B[3].a = B[0] = S[1], \qquad S[1].b = B[0].b = B[1] = S[2].$$

$InPath(0, 1, 4)$ возвращает *Good*.

Затем вызывается $InPath(0, 3, 4)$. Перебираются различные варианты множества $B[0]$. Пусть $B[0] = \{1, 3\}$, это будет $S[3 \cdot 2^{n-3}] = S[3]$. Вычисляются $r = 2, \ell = 1$. Находятся буквы $a = a_2, b = a_1$ такие, что

$$S[2].a = B[1].a = B[0] = S[3], \ S[3].b = B[0].b = B[2] = S[4].$$

Функция $InPath(0, 3, 4)$ возвращает *Good*, и $InPath(1, 2, 4)$ также возвращает *Good*. Мощность множества $B[2] = S[4]$ равна 1, поэтому $InPath(2, 4, 4)$ возвращает *Found*. Автомат оказался бережно синхронизируемым.

Докажем, что алгоритм $CheckLength(\mathscr{A}, L)$ работает правильно. Рассмотрим *дерево вызовов* Tr функции $InPath$. Вершинами в этом дереве являются вызовы $InPath(k, c, L)$, будем их для краткости обозначать через $IP(k, c)$ (L одно и то же для всех вызовов $InPath$). Ребро из вершины $IP(k_1, c_1)$ в вершину $IP(k_2, c_2)$ существует тогда и только тогда, когда функция $InPath(k_2, c_2, L)$ рекурсивно вызывается при выполнении функции $InPath(k_1, c_1, L)$. Вершины вида $IP(k, c)$ будем называть вершинами *k-го уровня*. Если $k > 0$, то при выполнении функции $InPath(k, c, L)$ происходит не более чем два рекурсивных вызова $InPath$: $InPath(k - 1, c - 2^{k-1}, L)$ и $InPath(k - 1, c + 2^{k-1}, L)$. Поэтому дерево Tr получится двоичным. Будем называть ребра вида $(IP(k, c), IP(k - 1, c - 2^{k-1}))$ *левыми*, а ребра вида $(IP(k, c), IP(k - 1, c + 2^{k-1}))$ *правыми*. Если $k = 0$, то функция $InPath$ далее рекурсивно не вызывается, поэтому дерево Tr состоит из уровней $\{n - 1, \ldots, 0\}$. Как в любом дереве, в Tr существует единственный путь из корня $IP(n - 1, 2^{n-1})$ до любой вершины.

Лемма 1. *Пусть $IP(k, c)$ – вершина в дереве Tr, $(c_{n-1}, \ldots, c_0)$ – двоичная запись числа c и $e_{n-1}, \ldots, e_{k+1}$ – ребра, составляющие путь из вершины $IP(n - 1, 2^{n-1})$ в вершину $IP(k, c)$, тогда для $i \in \{n - 1, \ldots, k + 1\}$*

$$c_i = \begin{cases} 0, & e_i \text{ – левое}, \\ 1, & e_i \text{ – правое}, \end{cases} \qquad c_k = 1, \qquad c_{k-1} = \ldots = c_0 = 0.$$

Доказательство. Докажем лемму индукцией по глубине вершины $IP(k,c)$. Для $k = n-1$ в дереве есть только одна вершина $IP(n-1, 2^{n-1})$. В этом случае $c = (1,0,\ldots,0)$, т. е. $c_k = 1$, $c_{k-1} = \ldots = c_0 = 0$. Пусть для вершин k-го уровня лемма доказана. Рассмотрим вершину v с $(k-1)$-го уровня дерева Tr. Она может иметь вид $IP(k-1, c-2^{k-1})$ или $IP(k-1, c+2^{k-1})$, где $IP(k,c)$ – вершина k-го уровня.

Если $v = IP(k-1, c-2^{k-1})$, тогда путь из корня дерева в вершину $IP(k,c)$ имеет вид $e_{n-1},\ldots,e_k$, где $e_k = (IP(k,c), IP(k-1, c-2^{k-1}))$, а $e_{n-1},\ldots,e_{k+1}$ – путь до вершины $IP(k,c)$. По предположению индукции $c = (c_{n-1},\ldots,c_{k+1},1,0,\ldots,0)$, где для $i \in \{n-1,\ldots,k+1\}$

$$c_i = \begin{cases} 0, & e_i \text{ – левое,} \\ 1, & e_i \text{ – правое.} \end{cases}$$

Пусть $(d_{n-1},\ldots,d_0)$ – двоичная запись числа $c - 2^{k-1}$. Тогда

$$(d_{n-1},\ldots,d_0) = (c_{n-1},\ldots,c_{k+1},0,1,0,\ldots,0).$$

Следовательно, $d_{k-1} = 1$, $d_{k-2} = \ldots = d_0 = 0$. Кроме того, ребро e_k левое, поэтому для $i \in \{n-1,\ldots,k\}$

$$d_i = \begin{cases} 0, & e_i \text{ – левое,} \\ 1, & e_i \text{ – правое.} \end{cases}$$

Аналогично рассматривается случай $v = IP(k-1, c+2^{k-1})$. Лемма доказана.

Лемма 2. *Для различных вершин $IP(k,c)$ дерева Tr, числа c различны, $1 \leqslant c \leqslant 2^n - 1$, и номер младшего ненулевого разряда в записи числа c равен k.*

Доказательство. К различным вершинам дерева Tr ведут различные пути из корня. Поэтому из леммы 1 следует, что для различных вершин $IP(k,c)$, числа c различны. Вершина $IP(k,c)$ содержится на k-м уровне, поэтому по лемме 1 двоичная запись числа c имеет вид $c = (c_{n-1},\ldots,c_{k+1},1,0,\ldots,0)$. Следовательно, номер младшего ненулевого разряда в записи числа c равен k. В двоичной записи числа c содержится n разрядов, поэтому $1 \leqslant c \leqslant 2^n - 1$. Лемма доказана.

Лемма 3. *Пусть $IP(0,c)$ – вершина 0-го уровня в дереве Tr; $e_{n-1},\ldots,e_1$ – ребра, составляющие путь из вершины $IP(n-1, 2^{n-1})$ в вершину $IP(0,c)$; ℓ = номер младшего ненулевого разряда в двоичной записи числа $c+1$;*

r = номер младшего ненулевого разряда в двоичной записи числа $c-1$. Тогда если $c < 2^n - 1$, то $IP(\ell, c+1)$ – начало последнего левого ребра в пути $e_{n-1}, \ldots, e_1$; если $c > 1$, то $IP(r, c-1)$ – начало последнего правого ребра в пути $e_{n-1}, \ldots, e_1$.

Доказательство. Из леммы 1 имеем, $c_0 = 1$, для $i \in \{n-1, \ldots, 1\}$

$$c_i = \begin{cases} 0, & e_i \text{ – левое}, \\ 1, & e_i \text{ – правое}. \end{cases}$$

Двоичная запись числа $c-1$ имеет вид

$$(c_{n-1}, \ldots, c_1, 0) = (c_{n-1}, \ldots, c_{r+1}, 1, 0, \ldots, 0).$$

Следовательно, r – номер последнего правого ребра в пути $e_{n-1}, \ldots, e_1$. Двоичная запись числа c имеет вид $(c_{n-1}, \ldots, c_{\ell+1}, 0, 1, \ldots, 1)$. Значит, двоичная запись числа $c+1$ имеет вид $(c_{n-1}, \ldots, c_{\ell+1}, 1, 0, \ldots, 0)$. Следовательно, ℓ – номер последнего левого ребра в пути $e_{n-1}, \ldots, e_1$. Лемма доказана.

Рассмотрим возможные варианты завершения работы $CheckLength(\mathscr{A}, L)$. Программа $CheckLength(\mathscr{A}, L)$ может завершить свою работу после того, как для некоторых k и c получилось $InPath(k, c, L) = Found$. В этом случае последовательно будет завершена обработка рекурсивных вызовов $InPath$, каждый из которых будет возвращать $Found$. В итоге $InPath(n-1, 2^{n-1}, L)$ так же вернет $Found$. В этом случае $CheckLength(\mathscr{A}, L)$ вернет ответ, что для автомата $\mathscr{A}$ существует б.с.с. длины, не превосходящей L.

Кроме того, программа $CheckLength(\mathscr{A}, L)$ может завершить свою работу после полного перебора всех возможных вариантов. В этом случае внутри единственного вызова $InPath(n-1, 2^{n-1}, L)$ будет совершено 2^n вызовов функции $InPath(n-2, 2^{n-2}, L)$ и 2^n вызовов функции $InPath(n-2, 3 \cdot 2^{n-2}, L)$, по одному для каждого множества T. Пусть $k \in \{n-1, \ldots, 1\}$, тогда внутри каждого вызова функции $InPath(k, c, L)$ будет совершено по 2^n вызовов $InPath(k-1, c-2^{k-1}, L)$ и $InPath(k-1, c+2^{k-1}, L)$. В случае полного перебора всех возможных вариантов $InPath(n-1, 2^{n-1}, L)$ вернет Bad. В этом случае $CheckLength(\mathscr{A}, L)$ вернет ответ, что для автомата $\mathscr{A}$ не существует б.с.с. длины, не превосходящей L.

Напомним, что вызовы $InPath(k, c, L)$ являются вершинами дерева вызовов Tr. Вызовы производятся в следующем порядке: после вызова функции $InPath(k, c, L)$ сначала производятся вызовы из левого поддерева с корнем $InPath(k, c, L)$ в дереве Tr, а затем вызовы из правого поддерева. Таким образом, при работе алгоритма фактически производится обход дерева Tr

в порядке корень, левое поддерево, правое поддерево. Такой обход дерева Tr является обходом дерева в глубину из вершины $IP(n-1, 2^{n-1})$, при котором в роли стека используется массив B. При вызове $InPath(k, c, L)$ вершиной стека является элемент $B[k]$, в который могут попасть различные множества $T \subseteq Q$. Нетрудно заметить, что стек содержит множества, получаемые при вызовах функции $InPath$, являющихся вершинами пути из корня дерева Tr в вершину $IP(k, c)$. То есть для $i \in \{n-1, k\}$ $B[i] = S[d_i]$, где $IP(n-1, c_{n-1}), \ldots, IP(k, c_k)$ – путь из вершины $IP(n-1, 2^{n-1})$ в вершину $IP(k, c)$. Для $k = 0$ при вызове $InPath(k, c, L)$ происходит проверка на существование букв a и b, и если подходящих букв не существует, то производится возврат поиска с уменьшением размера стека.

Выпишем все вершины дерева в порядке обхода, при котором не производится ни одного возврата: $IP(k_1, c_1), \ldots, IP(k_{2^n-1}, c_{2^n-1})$. Например, для $n = 3$ получится следующая последовательность:

$$IP(2,4), IP(1,2), IP(0,1), IP(0,3), IP(1,6), IP(0,5), IP(0,7).$$

По лемме 2 числа $c_1, \ldots, c_{2^n-1}$ различны и $1 \leqslant c_1, \ldots, c_{2^n-1} \leqslant 2^n - 1$. Поэтому найдется такое $t \in \{1, \ldots, 2^n - 1\}$, что $c_t = |w|$. При выполнении $InPath(k, c, L)$ производится перебор множеств $T \subseteq Q$. Пусть перебор множеств внутри любого вызова $InPath(k, c, L)$ начинается с множества T_0. Положим для всех $c > |w|$, $S[c] = T_0$.

Лемма 4. *Если существует б.с.с. w для автомата $\mathscr{A}$ и $|w| \leqslant L$, то алгоритм $CheckLength(\mathscr{A}, L)$ вернет положительный ответ.*

Доказательство. Положим $S[0] = Q$ и для $i \in \{1, \ldots, |w|\}$ $S[i] = Q.w[1, i]$. Множество $S[|w|]$ будет одноэлементным. Предположим от противного, что алгоритм $CheckLength(\mathscr{A}, L)$ вернет отрицательный ответ. В этом случае не один из вызовов $InPath$ не вернет $Found$.

При работе вызова $InPath(k_1, c_1, L) = InPath(n-1, 2^{n-1}, L)$ в некоторый момент в качестве множества T окажется множество $S[c_1] = S[2^{n-1}]$. При $T = S[2^{n-1}]$ будет произведен вызов функции $InPath(k_2, c_2, L) = = InPath(n-2, 2^{n-2}, L)$, в некоторый момент в качестве множества T окажется множество $S[c_2] = S[2^{n-2}]$ и т. д. Будут поочередно производиться вызовы $InPath(k_i, c_i)$, и каждый раз будет находиться множество $T = S[c_i]$. При этом множество $S[c_i]$ будет попадать в элемент $B[k_i]$.

Если c_i нечетное, то из леммы 2 получаем, что $k_i = 0$. В этом случае вычисляются числа ℓ и r такие, что по лемме 3 вершины $IP(\ell, c_i+1)$ и $IP(r, c_i-1)$ соответствуют началам последних левого и правого ребер в пути от корня дерева до вершины $IP(k_i, c_i)$. Массив B содержит множества, получаемые при

вызовах функции $InPath$, являющихся вершинами пути из корня дерева Tr в вершину $IP(k_i, c_i)$. Поэтому $S[c_i - 1] = B[r]$, $S[c_i + 1] = B[\ell]$. В этом случае производится поиск букв a и b таких, что $S[c_i - 1].a = S[c]$ и $S[c_i].b = S[c_i + 1]$. В качестве букв a и b подходят $w[c_i]$ и $w[c_i + 1]$, поэтому буквы a и b найдутся и $InPath(k_i, c_i)$ вернет $Good$.

Вызовы будут $InPath(k_i, c_i)$ продолжаться, пока c_i не станет равным $|w|$. В этом случае $|T| = |S[c_i]| = 1$ и $InPath(k_i, c_i)$ вернет $Found$. Затем последовательно будет завершена обработка рекурсивных вызовов $InPath$, каждый из которых будет возвращать $Found$. В итоге $InPath(n - 1, 2^{n-1}, L)$ также вернет $Found$. Мы пришли к противоречию. Лемма доказана.

Лемма 5. *Если алгоритм $CheckLength(\mathscr{A}, L)$ возвращает положительный ответ, то существует б.с.с. для автомата $\mathscr{A}$ длины, не большей чем L.*

Доказательство. Некоторый вызов функции $InPath$ вернет $Found$. Пусть $InPath(k, c, L)$ – первый из таких вызовов. Множество T, найденное при вызове $InPath(k, c, L)$, будет одноэлементным и c не будет превосходить L. Напомним, что $IP(k_1, c_1), \ldots, IP(k_{2^n-1}, c_{2^n-1})$ – порядок обхода вершин дерева Tr алгоритмом. Для некоторого $i \in \{1, \ldots, 2^n - 1\}$, $k_i = k, c_i = c$, причем для любого $d \in \{1, \ldots, c_i\}$ найдется такое j, что вершину $IP(j, d)$ алгоритм к моменту обхода $IP(k, c)$ уже обошел.

Построим б.с.с. w для автомата $\mathscr{A}$. Пусть $d \in \{c_1, \ldots, c_i\}$. В дереве Tr существует вершина $IP(j, d)$. При последнем вызове находится некоторое множество T, которое помещается а $B[j]$. Положим $S[d] = T$. Положим также $S[0] = Q$. Пусть d – произвольное нечетное число от 1 до c, в этом случае $d \in \{c_1, \ldots, c_i\}$, и в дереве Tr есть вершина $IP(0, d)$. При последнем вызове $InPath(0, d, L)$ производится поиск букв a и b таких, что $B[r].a = T$ и $T.b = B[\ell]$. Если буквы a и b не найдутся, то произойдет возврат поиска и вызов $InPath(0, d, L)$ не будет последним. Поэтому буквы a и b найдутся. Положим $w[d] = a$, $w[d + 1] = b$. Тем самым мы определили слово w.

Докажем, что для всех $d \in \{1, \ldots, c\}$ $Q.w[1, d] = S[d]$. Имеем $S[0] = Q$. Пусть $Q.w[1, d - 1] = S[d - 1]$. Если d – нечетное, то в дереве Tr существует вершина $IP(j, d)$ и при последнем вызове $InPath(0, d, L)$ находятся буквы a и b такие, что $B[r].a = T$ и $T.b = B[\ell]$. Повторяя рассуждения из доказательства леммы 4, получим, что $S[d - 1] = B[r]$, $S[d + 1] = B[\ell]$. Следовательно, $S[d] = B[r].a = S[d - 1].w[d] = Q.w[1, d]$. Случай четного d доказывается аналогично рассмотрением вызова $InPath(0, d - 1, L)$. Следовательно, $|Q.w[1, c]| = |S[c]| = 1$ и $|w| = c \leqslant L$. Лемма доказана.

Из лемм 4 и 5 следует, что алгоритм $CheckLength(\mathscr{A}, L)$ возвращает положительный ответ тогда и только тогда, когда существует б.с.с. для автомата

$\mathscr{A}$ длины, не большей чем L. Алгоритм $CheckLength(\mathscr{A}, L)$ использует дополнительную память полиномиального размера, а именно $\Theta(n^2)$, поэтому задача БЕР СИНХ ДЛИНЫ L лежит в классе PSPACE.

При помощи функции $InPath$ могут быть также решены задачи БЕР СИНХ и КРАТ БЕР СИНХ. Если ЧКА $\mathscr{A}$ с n состояниями бережно синхронизируем, то, как уже упоминалось выше, существует б.с.с. для $\mathscr{A}$ длины не более $2^n - 2$. Поэтому проверку заданного автомата $\mathscr{A}$ на бережную синхронизируемость, а значит и решение задачи БЕР СИНХ можно осуществлять следующим образом:

Функция $CheckSynchronizing(\mathscr{A})$
$B[n] = Q$
Если $InPath(n-1, 2^{n-1}, 2^n - 2) = Found$, **то**
    Вернуть Автомат $\mathscr{A}$ бережно синхронизируем
Иначе
    Вернуть Автомат $\mathscr{A}$ не является бережно синхронизируемым
Конец функции

Алгоритм $CheckSynchronizing(\mathscr{A})$ использует дополнительную память полиномиального размера, следовательно, задача БЕР СИНХ лежит в классе PSPACE.

Для заданного ЧКА $\mathscr{A}$ длину кратчайшего б.с.с. можно вычислить, используя двоичный поиск по всем возможным длинам от 1 до $2^n - 2$, при каждом выборе интервала для поиска запуская $InPath$. Для заданных ЧКА $\mathscr{A}$ и целого числа L проверить, имеет ли кратчайшее б.с.с. для автомата $\mathscr{A}$ длину L, а значит и решить задачу КРАТ БЕР СИНХ можно при помощи следующей программы:

Функция $CheckShortest(\mathscr{A}, L)$
Если $InPath(n-1, 2^{n-1}, 2^n - 2) = Bad$, **то**
    Вернуть Автомат $\mathscr{A}$ не является бережно синхронизируемым
Иначе
    $mi = 1$
    $ma = 2^n - 2$
    Пока $ma > mi$
        $len = \lfloor (ma + mi)/2 \rfloor$
        $B[n] = Q$
        Если $InPath(n-1, 2^{n-1}, len) = Found$, **то**
            $ma = len$
        Иначе
            $mi = len + 1$

Если $ma = L$, **то**
Вернуть Длина кратчайшего б.с.с. для $\mathscr{A}$ равна L
Иначе
Вернуть Длина кратчайшего б.с.с. для $\mathscr{A}$ не равна L
Конец функции

Алгоритм $CheckShortest(\mathscr{A}, L)$ использует дополнительную память полиномиального размера, следовательно, задача КРАТ БЕР СИНХ лежит в классе PSPACE. Таким образом, предложение доказано.

4. PSPACE-трудность

Предложение 3. *Задача* 2-БЕР СИНХ PSPACE-*трудна.*

Доказательство. Сведем к задаче 2-БЕР СИНХ задачу 3-ВЫП С КВАНТОРАМИ. Пусть на вход задачи 3-ВЫП С КВАНТОРАМИ подана формула

$$F = Q_1 x_1 Q_2 x_2 \dots Q_n x_n P(x_1 \dots x_n)$$

такая, что для любого $i \in \{1, \dots, n\}$ $Q_i \in \{\exists, \forall\}$, x_i – булева переменная; $P(x_1 \dots x_n)$ – булева формула в конъюнктивной нормальной форме вида $\bigwedge\limits_{j=1}^{N} (y_1^j \vee y_2^j \vee y_3^j)$, $N \geqslant 1$, где $y_1^j, y_2^j, y_3^j \in \{x_i | i \in \{1 \dots n\}\} \cup \{\neg x_i | i \in \{1 \dots n\}\}$, $n \geqslant 1$. Переменные x_i, а также результаты вычисления операций $\neg, \wedge, \vee$, функции $P(x_1, \dots, x_n)$ и формулы F могут принимать значения 0 и 1. Задача 3-ВЫП С КВАНТОРАМИ PSPACE-полна (см. [12]).

Мы будем предполагать, что в формуле F присутствует хотя бы один квантор всеобщности и хотя бы один квантор существования. Задача 3-ВЫП С КВАНТОРАМИ остается PSPACE полной в этом случае. К тому же если кванторов всеобщности или существования в формуле нет, то рассуждения могут быть проведены аналогичным образом.

4.1. Алгоритм проверки на выполнимость

Сначала мы опишем алгоритм, проверяющий выполнимость формулы F. Затем построим ЧКА, процесс бережной синхронизации которого будет соответствовать описанному алгоритму. Алгоритм будет использовать оракула, который в любой момент может подсказать, какое значение следует придать очередной переменной, чтобы в конечном счете формула F стала истинной. Если подходят оба значения переменной или не подходит ни одно, то оракул выбирает значение случайным образом. Алгоритм будет представлять собой

рекурсивный вызов функции $SAT(k)$, которая, в свою очередь, делает следующее:

Функция $SAT(k)$
Если $k \leqslant n$, **то**
    Если $Q_k = \exists$, **то**
        Спросить у оракула значение переменой x_k
        Вызвать $SAT(k+1)$
    Если $Q_k = \forall$, **то**
        Положить $x_k = 0$
        Вызвать $SAT(k+1)$
        Положить $x_k = 1$
        Вызвать $SAT(k+1)$
Иначе
    Если $P(x_1, \ldots, x_n) = 0$, **то** $f = 0$
Конец функции

Переменная f является глобальной. Для проверки формулы на выполнимость следует положить $f = 1$, а затем вызвать $SAT(1)$. Если после окончания работы алгоритма значение переменной f по-прежнему равно 1, то формула F истинна. Если в какой-то момент значение переменной f стало равным 0, то формула F ложна. В дальнейшем мы будем ссылаться на только что описанный алгоритм как на $SAT(1)$.

Алгоритм работает следующим образом. Каждой переменной x_i, соответствующей квантору всеобщности $Q_i = \forall$, сначала присваивается значение $x_i = 0$, а затем $x_i = 1$. Пусть $Q_{i+1} = \ldots = Q_{t-1} = \exists$. Тогда для каждого значения переменной x_i переменным $x_{i+1}, \ldots, x_{t-1}$ должны быть присвоены конкретные значения. Эти значения мы запрашиваем у оракула. Таким образом, для каждого значения переменной x_i мы получаем фиксированный набор значений переменных $x_{i+1}, \ldots, x_{t-1}$. Для каждого такого набора значений мы перебираем все возможные (0 или 1) значения переменной x_t и т. д. Таким образом, мы переберем все комбинации возможных значений переменных, соответствующих кванторам всеобщности. Для каждой такой комбинации будут указаны значения переменных, соответствующих кванторам существования, и посчитано значение функции $P(x_1, \ldots, x_n)$. Если для всех комбинаций выполняется $P(x_1, \ldots, x_n) = 1$, то формула F истинна. Если для какого-то набора значений $x_1, \ldots, x_n$, $P(x_1, \ldots, x_n) = 0$, то в какой-то момент оракул не смог указать подходящее значение переменной и указал случайное.

Для дальнейшего использования приведенный выше алгоритм удобно описать на языке двоичного счетчика возможных комбинаций значений пе-

ременных, перед которыми стоят кванторы всеобщности. Пусть во множестве $\{Q_i \mid i = 1 \dots n\}$ содержится λ кванторов всеобщности и пусть $h(1) < h(2) < \dots < h(\lambda)$ – номера кванторов всеобщности в последовательности $Q_1, \dots, Q_n$. Назовем переменные $x_{h(1)}, \dots, x_{h(\lambda)}$ *переменными всеобщности*. Остальные переменные назовем *переменными существования*.

Каждая из переменных $x_{h(1)}$ может принимать значения 0 и 1. Поэтому значения переменным $x_{h(1)}, \dots, x_{h(\lambda)}$ могут быть присвоены 2^λ различными способами. Введем *счетчик* C всевозможных комбинаций значений переменных всеобщности. Пусть C – целая переменная, принимающая значения от 0 до $2^\lambda - 1$. Двоичная запись числа C будет представлять собой выписанные подряд значения переменных $x_{h(1)}, \dots, x_{h(\lambda)}$, причем младшему биту в записи будет соответствовать переменная $x_{h(\lambda)}$. Мы будем записывать $C = (x_{h(1)}, \dots, x_{h(\lambda)})$.

Проследим за работой алгоритма $SAT(1)$ и поведением счетчика C на каждом его шаге. В начале работы алгоритма происходит последовательный вызов функций $SAT(k)$ для всех k от 1 до $n+1$, назовем это *нулевым шагом* работы алгоритма. В конце нулевого шага работы алгоритма, при первом вызове $SAT(n+1)$, значение счетчика C равно $(0, \dots, 0) = 0$.

Разобьем дальнейшую работу алгоритма на $2^\lambda - 1$ *шагов*. Пусть номер шага $M \in \{1, \dots, 2^\lambda - 1\}$, и к началу M-го шага алгоритма:

- значение счетчика $C = M - 1$;
- алгоритм завершил обработку очередного, M-го по счету вызова функции $SAT(n+1)$.

Заметим, что эти условия выполняются на момент начала первого шага алгоритма, т. е. для $M = 1$. Положим $k_0 = \max\{k \in \{1, \dots, n\} \mid Q_k = \forall \text{ и } x_k = 0\}$.

M-й шаг алгоритма состоит в следующем:

1. Производится поочередное завершение работы функций $SAT(k)$ – для всех k от $k = n+1$ до $k_0 + 1$.
2. Продолжается выполнение функции $SAT(k_0)$ и присваивается $x_{k_0} = 1$.
3. Выполняется поочередный вызов функций $SAT(k)$ для $k = k_0+1, \dots, n$. При вызове функции $SAT(k)$, если $Q_k = \exists$, то производится выбор значения переменной существования x_k при помощи оракула, а если $Q_k = \forall$, то производится присвоение $x_k = 0$. После этого вызывается $SAT(k+1)$.
4. Вызывается $SAT(n+1)$ и вычисляется значение функции $P(x_1, \dots, x_n)$.

Пусть в записи числа $M-1$ номер младшего нулевого бита равен τ. Тогда для любого $\ell \in \{\tau+1, \lambda\}$, $x_{h(\ell)} = 1$. В то же время $x_{h(\tau)} = 0$ и, следовательно, $k_0 = h(\tau)$. После M-го шага алгоритма значение переменной $x_{h(\tau)}$ становится равным 1, для каждого $\ell \in \{\tau+1, \lambda\}$ значение переменной $x_{h(\ell)}$ становится равным 0, для всех $\ell \in \{1, \tau-1\}$ значения переменных $x_{h(\ell)}$ не изменяются. Следовательно, значение счетчика C изменяется с $(x_{h(1)}, \ldots, x_{h(\tau-1)}, 0, 1, \ldots, 1)$ на $(x_{h(1)}, \ldots, x_{h(\tau-1)}, 1, 0, \ldots, 0)$, т. е. к C прибавляется 1. Таким образом, значение переменной C становится равным M.

После M-го шага алгоритма имеем:

- значение счетчика $C = M$;
- алгоритм завершил обработку очередного, $(M+1)$-го по счету вызова функции $SAT(n+1)$.

Следовательно, понятие шага алгоритма определено корректно. Заметим, что на M-м шаге могут измениться значения переменных существования с номерами, большими чем k_0.

Обобщим сказанное выше. На каждом из 2^λ шагов алгоритма (от 0 до $2^\lambda - 1$) устанавливается очередной набор значений переменных всеобщности, определяются значения соответствующих переменных существования и вычисляется значение функции $P(x_1, \ldots, x_n)$ при установленных значениях переменных. Если при всех комбинациях значений переменных всеобщности получается $P(x_1, \ldots, x_n) = 1$, то формула F объявляется истинной.

4.2. Построение множества состояний автомата

Построим двухбуквенный ЧКА $\mathscr{A} = \langle Q, \Sigma, \delta \rangle$, процесс бережной синхронизации которого моделирует вышеописанный алгоритм $SAT(1)$, проверяющий формулу $F = Q_1 x_1 Q_2 x_2 \ldots Q_n x_n P(x_1 \ldots x_n)$ на истинность. Положим $\Sigma = \{a, b\}$. Пусть функция $P(x_1 \ldots x_n)$ является композицией p операций $\wedge, \vee$ и $\neg$. Для вычисления значения функции $P(x_1 \ldots x_n)$ необходимо последовательно применить эти операции. Пусть при вычислении функции P эти операции выполняются в порядке $f_1, \ldots, f_p$, где $f_i \in \{\wedge, \vee, \neg\}$. Заметим, что при вычислении функции P операции $\neg$ применяются только непосредственно к переменным. Поэтому можно считать, что сначала выполняются все операции $\neg$, а потом все остальные.

Пусть среди операций $f_1, \ldots, f_p$ операция $\neg$ встречается p_1 раз. Тогда $f_1 = \ldots = f_{p_1} = \neg$, $f_{p_1+1}, \ldots, f_p \in \{\wedge, \vee\}$. Обозначим $s = n + 2p_1 + 4(p - p_1)$. Определим множество состояний автомата $\mathscr{A}$. Положим $Q = T_v \cup T_q \cup \{end\}$,

где

$$\begin{aligned} T_v =& \{v(m,i,k) \mid m = 0 \dots s+1, i = 1 \dots n, k \in \{0,1\}\}, \\ T_q =& \{q(m,i,S) \mid m \in 0 \dots n, i \in 1 \dots p-1\} \cup \\ & \cup \{q(m,p,k) \mid m \in 0 \dots n, k \in \{N,S\}\} \cup \\ & \cup \{q(m,i,k) \mid m \in n+1 \dots s+1, i \in 1 \dots p, k \in \{0,1,U\}\}. \end{aligned}$$

Таким образом, состояния $T_v \cup T_q$ могут быть изображены в виде таблицы из $s+2$ рядов (от 0 до $s+1$) и $n+p$ колонок (n колонок для состояний $v(m,i,k)$, мы их будем называть *колонками переменных*, и p колонок для состояний $q(m,i,k)$, мы их будем называть *колонками операций*). В каждой клеточке таблицы изображены все состояния $q(m,i,k)$ или все состояния $v(m,i,k)$ для некоторых фиксированных i и m и всех возможных k. Состояние end, в которое будет синхронизироваться автомат, удобно представлять расположенным вне таблицы. Пример такого автомата для формулы

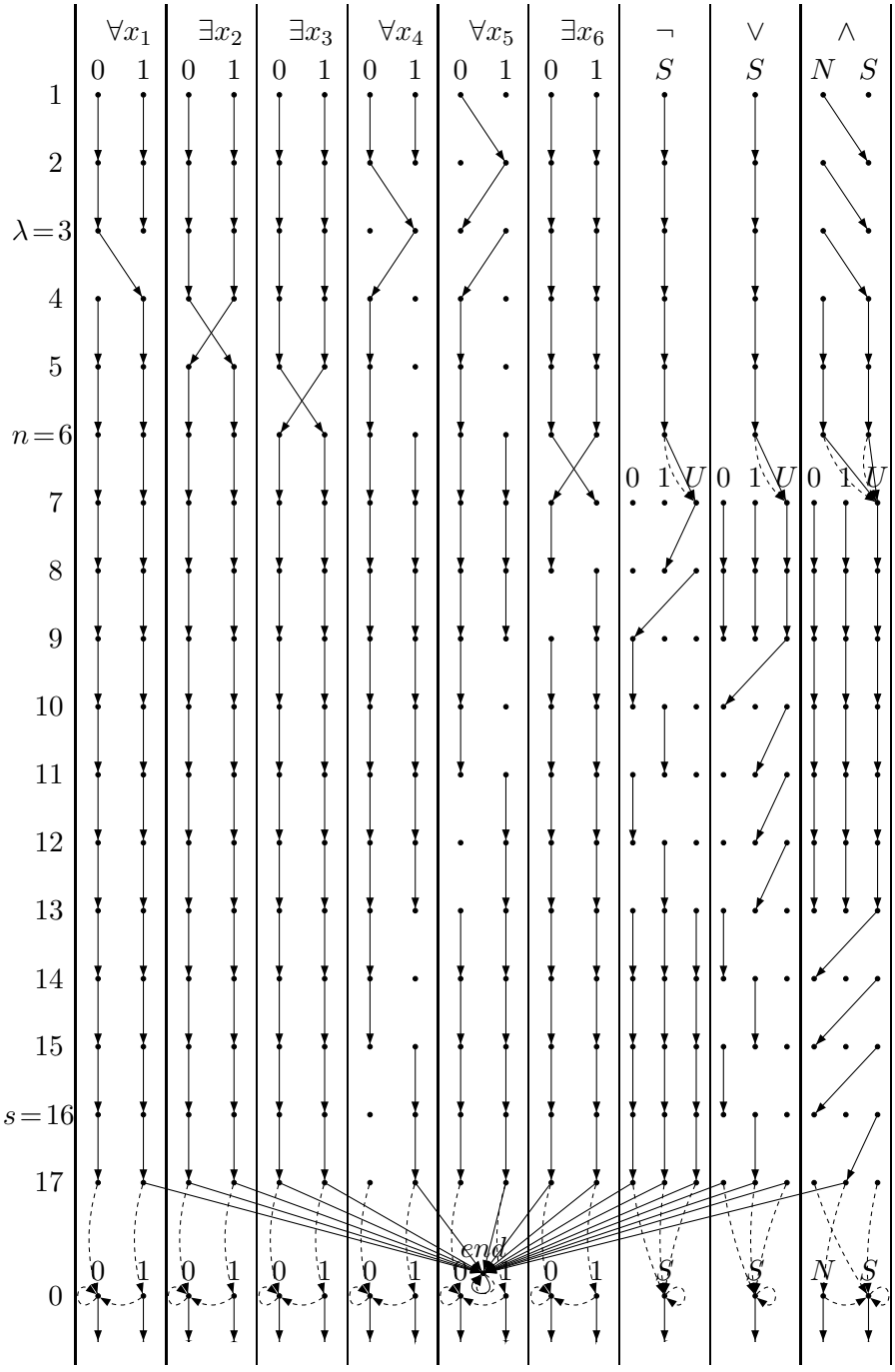
$$\forall\, x_1\ \exists\, x_2\ \exists\, x_3\ \forall\, x_4\ \forall\, x_5\ \exists\, x_6\ (\neg\, x_6 \vee x_5)\ \wedge x_4$$

изображен на рисунке. Сплошными линиями изображено действие буквы a. Буква a определена не на всех состояниях автомата, а буква b на всех. Чтобы не загромождать рисунок, действие буквы b изображено не для всех состояний. На рисунке для каждого состояния либо действие буквы b изображено пунктирными линиями, либо буква b просто переводит состояния в следующий ряд, т. е. состояние $v(m,i,k)$ переводит в состояние $v(m+1,i,k)$, а состояние $q(m,i,k)$ – в состояние $q(m+1,i,k)$. Стрелки вниз из состояний 0-го ряда (на рисунке нижнего) означают переход в 1-й (верхний) ряд под действием буквы a. Для автомата, изображенного на рисунке, $n = 6$, $\lambda = 3$, $p = 3$, $p_1 = 1$, $s = 16$.

4.3. Описание работы автомата

Перед тем как формально вводить функцию переходов автомата, опишем вкратце механизм его действия. Удобнее всего представлять себе бережную синхронизацию автомата $\mathscr{A}$ на языке фишек. Пусть в начале на каждом из состояний автомата стоит по фишке. Под действием любой буквы любая фишка, стоящая на состоянии i-го ряда, будет переходить в $(i+1)$-й ряд (в некоторых случаях это будет не так). Фишка из состояния $(s+1)$-го ряда будет переходить в 0-й ряд. Состояния $v(m,i,k)$(колонка с номером i) для разных m и k соответствуют переменной x_i. Причем фишка, стоящая на состоянии $v(m,i,0)$, соответствует значению $x_i = 0$, а фишка, стоящая на состоянии $v(m,i,1)$, соответствует значению $x_i = 1$.

В дальнейшем мы опишем структуру произвольного кратчайшего б.с.с. для ЧКА $\mathscr{A}$. Кратчайшее б.с.с. для ЧКА $\mathscr{A}$ будет состоять из двух частей –



Автомат для формулы $\forall x_1\, \exists x_2\, \exists x_3\, \forall x_4\, \forall x_5\, \exists x_6\, (\neg x_6 \vee x_5) \wedge x_4$

u' и u. Слово u' переводит множество всех состояний автомата, кроме состояния end, в подмножество 0-го ряда. Причем для всех $i \in \{1, \ldots, n\}$ выполняется $v(0, i, 0) \in \delta(Q, u')$, $v(0, i, 1) \notin \delta(Q, u')$, что соответствует значениям $x_i = 0$ для всех $i \in \{1, \ldots, n\}$, в частности для $i \in \{h(1), \ldots, h(\lambda)\}$. Таким образом, применение слова u' моделирует подготовку к работе и обнуление счетчика $C = (x_{h(1)}, \ldots, x_{h(\lambda)})$ в алгоритме $SAT(1)$. Слово u состоит из 2^λ частей $u_0, \ldots, u_{2^\lambda - 1}$. Для любого $M \in \{0, \ldots, 2^\lambda - 1\}$ слово u_M моделирует M-й шаг алгоритма $SAT(1)$. В начале M-го шага все оставшиеся фишки находятся в состояниях 0-го ряда. Слово u_M имеет длину $s + 2$ и при побуквенном применении переводит фишки последовательно из 0-го ряда в 1-й, затем во 2-й и т. д., в $(s + 1)$-й и затем обратно в 0-й. Ряды состояний с номерами от 1 до n моделируют присвоение значений переменным, причем присвоение значений переменным всеобщности моделируется рядами с номерами от 1 до λ. После применения очередного слова u_M значение счетчика C становится равным M.

Ряды с номерами $n + 1, \ldots, s$ соответствуют последовательному вычислению значений операций $f_1, \ldots, f_p$. Каждому отрицанию ($\neg$) соответствует по два ряда, а каждой дизъюнкции($\vee$) и конъюнкции($\wedge$) – по четыре ряда. При этом состояния $q(m, i, k)$ для разных m и k (колонка с номером i) соответствуют операции f_i. Причем фишка на состоянии $q(m, i, 0)$ соответствует значению $f_i(\cdot) = 0$; фишка на состоянии $q(m, i, 1)$ – значению $f_i(\cdot) = 1$; фишка на состоянии $q(m, i, U)$ означает, что на текущем шаге значения переменных уже определены, а значение функции $f_i(\cdot)$ еще не посчитано, фишка на состоянии $q(m, i, N)$ (такие будут только в последней колонке) означает, что значения переменных $x_{h(1)}, \ldots, x_{h(\lambda)}$ на текущем шаге еще не определены и могут изменяться; фишка на состоянии $q(m, i, S)$ в последней колонке означает, что значения переменных $x_{h(1)}, \ldots, x_{h(\lambda)}$ на текущем шаге уже изменяться не будут.

Ряды с номерами $s + 1$ и 0 будут моделировать проверку вычисленного значения на истинность ($P(x_1, \ldots, x_n) = 1$). В конце каждого шага результаты всех операций должны быть посчитаны, и значение последней операции f_p должно получиться равным 1 (чтобы значение всей функции $P(x_1, \ldots, x_n)$ было истинным). Если все операции на данном шаге алгоритма выполнились и $P(x_1, \ldots, x_n) = 1$, то алгоритм, а вместе с ним и бережная синхронизация автомата переходит на следующий шаг (применение слова u_{M+1}). Если все шаги алгоритма были успешно пройдены, то в конце $(2^\lambda - 1)$-го шага появится возможность перевести все фишки из $(s + 1)$-го ряда в состояние end и тем самым завершить бережную синхронизацию автомата $\mathscr{A}$. Если в алгоритме $SAT(1)$ в конце какого-нибудь шага значение $P(x_1, \ldots, x_n)$ получилось равным 0 (это означает, что оракул не смог указать правильное значение

какой-то переменной и выбрал случайное), то автомат не сможет перейти на следующий шаг бережной синхронизации и дальше изменять значение счетчика C. В этом случае не будет существовать б.с.с. для построенного автомата. Таким образом, длина кратчайшего б.с.с. для автомата $\mathscr{A}$ будет равна $(2^\lambda + 1)(s + 2)$.

В примере на рисунке присвоение значений переменным всеобщности x_5, x_4 и x_1 производится в 1, 2 и 3-м ряду соответственно. Присвоение значений переменным существования x_2, x_3 и x_6 производится в 4, 5 и 6-м ряду соответственно; 7-й и 8-й ряды соответствуют вычислению значения выражения $\neg\, x_6$ (операция $\neg$); 9–12-й ряды соответствуют вычислению значения выражения $\neg\, x_6 \vee x_5$ (операция $\vee$); 13–16-й ряды соответствуют вычислению значения выражения $P(x_1, x_2, x_3, x_4, x_5, x_6) = (\neg\, x_6 \vee x_5) \wedge x_4$ (операция $\wedge$); 17-й ряд соответствует проверке вычисленного значения функции $P(x_1, x_2, x_3, x_4, x_5, x_6)$ на истинность. В случае если $P(x_1, \ldots, x_6) = 1$, шаг завершается и автомат переходит в состояние готовности к следующему изменению счетчика.

4.4. Задание функции переходов

Определим функцию δ формально. Как и прежде, мы будем обозначать $q.\alpha = \delta(q, \alpha)$ для $q \in Q, \alpha \in \Sigma$. Если значение $q.\alpha$ не определено, мы будем писать $q.\alpha = undef$.

При определении действия функции δ мы даем некоторые иллюстрирующие пояснения. Они касаются передвижения фишек по автомату в тех случаях, когда все имеющиеся фишки располагаются в одном ряду по одной фишке в каждой колонке. В частности, это означает, что определены значения всех переменных, а значит и значение счетчика.

Переходы, соответствующие присвоению значений переменным

Определим порядок присвоения значений переменным. Для этого определим функцию $\sigma(m)$, $m = 1 \ldots n$. Значение $\sigma(m)$ будет указывать на номер переменной, присвоение значения которой моделируется m-м рядом состояний в автомате $\mathscr{A}$. Соответственно значение $\sigma^{-1}(i),\ i \in \{1 \ldots n\}$ будет указывать на номер ряда, в котором моделируется присвоение значения переменной x_i. Напомним, что $h(1), \ldots, h(\lambda)$ – номера переменных всеобщности. Положим $\sigma(1) = h(\lambda)$, $\sigma(2) = h(\lambda - 1), \ldots, \sigma(\lambda) = h(1)$. Ряд состояний с номером i будет соответствовать квантору $Q_{\sigma(i)}$. В примере (см. рисунок) $\sigma(1) = 5$, $\sigma(2) = 4$, $\sigma(3) = 1$.

Пусть $m \in \{1, \ldots, \lambda\}$ и, следовательно, $Q_{\sigma(m)} = \forall$. Положим:

для $i \in \{1, \ldots, \sigma(m) - 1\}$

$v(m, i, 0).a = v(m + 1, i, 0)$, $v(m, i, 1).a = v(m + 1, i, 1)$,

$v(m, \sigma(m), 0).a = v(m + 1, \sigma(m), 1)$, $v(m, \sigma(m), 1).a = undef$;

для $i \in \{\sigma(m) + 1, \ldots, n\}$ $v(m, i, 0).a = \begin{cases} undef, & Q_i = \forall, \\ v(m + 1, i, 0), & Q_i = \exists; \end{cases}$

для $i \in \{\sigma(m) + 1, \ldots, n\}$ $v(m, i, 1).a = \begin{cases} v(m + 1, i, 0), & Q_i = \forall, \\ v(m + 1, i, 1), & Q_i = \exists. \end{cases}$

Таким образом, если в процессе синхронизации двоичная запись текущего значения счетчика имеет вид $(\ldots \ \ldots \underset{t}{0}\, 1 \ldots 1)$, где $h(t) = \sigma(m)$, то при прохождении фишками рядов $1, \ldots, \lambda$ букву a можно применить только тогда, когда фишки будут в ряду с номером m. Если в этот момент буква a будет применена, то счетчик примет значение $(\ldots \ \ldots \underset{t}{1}\, 0 \ldots 0)$.

Для $i \in \{1, \ldots, n\}$ $v(m, i, 0).b = v(m + 1, i, 0)$, $v(m, i, 1).b = v(m + 1, i, 1)$.

Буква b просто переводит фишки в следующий ряд.

Для $i \in \{1, \ldots, p - 1\}$ $q(m, i, S).a = q(m + 1, i, S)$,

$q(m, p, N).a = q(m + 1, p, S)$, $q(m, p, S).a = undef$.

Букву a можно применять (а соответственно и изменять значения счетчика), только если фишка из последней колонки находится в состоянии $q(m, p, N)$. На каждом шаге при прохождении фишками рядов $1, \ldots, \lambda$ буква a может быть применена не более одного раза, так как после этого фишка из последней колонки переходит в состояние $q(m, p, S)$.

Для $i \in \{1, \ldots, p\}$, $q(m, i, S).b = q(m+1, i, S)$, $q(m, p, N).b = q(m+1, p, N)$.

Буква b просто переводит фишки в следующий ряд.

Пусть $g(1) < g(2) < \ldots < g(n - \lambda)$ – номера переменных существования. Положим $\sigma(\lambda + 1) = g(1)$, $\sigma(\lambda + 2) = g(2), \ldots, \sigma(n) = g(n - \lambda)$. В примере на рисунке $\sigma(4) = 2$, $\sigma(5) = 3$, $\sigma(6) = 6$.

Пусть $m \in \{\lambda+1, \ldots, n\}$ и, следовательно, $Q_{\sigma(m)} = \exists$. Положим:

для $i \in \{1, \ldots, \sigma(m)-1\}$

$v(m,i,0).a = v(m+1,i,0)$, $v(m,i,1).a = v(m+1,i,1)$,

$v(m,\sigma(m),0).a = v(m+1,\sigma(m),1)$, $v(m,\sigma(m),1).a = v(m+1,\sigma(m),0)$;

для $i \in \{\sigma(m)+1, \ldots, n\}$ $v(m,i,0).a = v(m+1,i,0)$;

для $i \in \{\sigma(m)+1, \ldots, n\}$ $v(m,i,1).a = \begin{cases} undef, & Q_i = \forall \\ v(m+1,i,1), & Q_i = \exists; \end{cases}$

для $i \in \{1, \ldots, n\}$ $v(m,i,0).b = v(m+1,i,0)$, $v(m,i,1).b = v(m+1,i,1)$.

Если значение, которое нужно присвоить переменной существования $x_{\sigma(m)}$ отличается от ее значения на предыдущем шаге, то при прохождении фишками m-го ряда нужно применять букву a, иначе нужно применять букву b, которая просто переводит фишки в следующий ряд. Причем значение переменной существования можно изменять (применением буквы a), только если непосредственно перед этим (на этом же шаге) было изменено значение одной из предшествующих переменных всеобщности (при изменении значения счетчика). В этом случае значения всех последующих переменных всеобщности равно 0, т. е. все фишки в колонках переменных всеобщности с номерами, большими чем $\sigma(m)$, находятся в состояниях $v(m,i,0)$.

Если $m \in \{1, \ldots, n-1\}$, то

для $i \in \{1, \ldots, p-1\}$ $q(m,i,S).a = q(m+1,i,S)$,

$q(m,p,N).b = q(m+1,p,N)$, $q(m,p,S).b = q(m+1,p,S)$.

Если $m = n$, то

для $i \in \{1, \ldots, p\}$, $q(m,i,S).a = q(m,i,S).b = q(m+1,i,U)$,

$q(m,p,N).b = q(m,p,N).a = q(m+1,p,U)$.

Если $m \in \{1, \ldots, n-1\}$, то буквы a и b просто переводят фишки в следующий ряд. Если $m = n$, то буквы a и b переводят фишки в состояния $q(n+1,i,U)$.

Переходы, соответствующие вычислению значений функции $P(x_1, \ldots, x_n)$

Напомним, что при вычислении значения функции $P(x_1, \ldots, x_n)$ сначала вычисляются значения p_1 операций $\neg$, а затем – значения $p - p_1$ операций с двумя аргументами ($\wedge$ или $\vee$). Вычисление каждой операции моделируется однократным применением буквы a в нужный момент.

Сначала определим действие функции δ на рядах состояний с номерами $n+1,\ldots,n+2p_1$, т. е. на рядах, моделирующих вычисление результатов операций $\neg$. Пусть $m \in \{1,\ldots,p_1\}$ и операция $f_m = \neg$ применяется к переменной x_k, $k \in \{1,\ldots,n\}$. Операции f_m соответствуют ряды с номерами $n+2m-1$ и $n+2m$. Положим:

для $i \in \{1,\ldots,n\},\ j \in \{0,1\}$	$v(n+2m-1,i,j).b = v(n+2m,i,j),$
	$v(n+2m,i,j).b = v(n+2m+1,i,j);$
для $i \in \{1,\ldots,n\}\backslash\{k\},\ j \in \{0,1\}$	$v(n+2m-1,i,j).a = v(n+2m,i,j),$
	$v(n+2m,i,j).a = v(n+2m+1,i,j);$
для $i \in \{1,\ldots,p\},\ j \in \{0,1,U\}$	$q(n+2m-1,i,j).b = q(n+2m,i,j),$
	$q(n+2m,i,j).b = q(n+2m+1,i,j);$
для $i \in \{1,\ldots,p\}\backslash\{m\},\ j \in \{0,1,U\}$	$q(n+2m-1,i,j).a = q(n+2m,i,j),$
	$q(n+2m,i,j).a = q(n+2m+1,i,j).$

Другими словами, операция (а соответственно и буква a) не использует и не меняет значения переменных и результатов операций кроме себя и своего аргумента, а так же, как и буква b, просто переводит фишки в следующий ряд, сохраняя их расположение в ряду.

Смоделируем действие операции $\neg$:

$$
\begin{aligned}
&v(n+2m-1,k,0).a = v(n+2m,k,0), v(n+2m-1,k,1).a = undef,\\
&v(n+2m,k,0).a = undef, v(n+2m,k,1).a = v(n+2m+1,k,1),\\
&q(n+2m-1,m,0).a = q(n+2m-1,m,1).a = undef,\\
&q(n+2m-1,m,U).a = q(n+2m,m,1),\\
&q(n+2m,m,0).a = q(n+2m,m,1).a = undef,\\
&q(n+2m,m,U).a = q(n+2m+1,m,0).
\end{aligned}
$$

Если аргумент операции f_m равен 0, то вычисление значения f_m производится на $(n+2m-1)$-м шаге, если аргумент операции f_m равен 1, то на $(n+2m)$-м шаге. Если результат вычисления равен 0, то после вычисления фишка находится на состоянии $q(n+2m+1,m,0)$. Если результат вычисления равен 1, то после вычисления фишка находится на состоянии $q(n+2m+1,m,1)$. Если вычисление значения f_m не было произведено, то фишка находится на состоянии $q(n+2m+1,m,U)$. В примере на рисунке вычисление операции $\neg$ производится в 7-м и 8-м рядах.

Теперь определим действие функции δ на рядах состояний с номерами $n+2p_1+1,\ldots,s$. Пусть $m \in \{p_1+1,\ldots,p\}$. В этом случае операция

$f_m \in \{\wedge, \vee\}$ имеет два аргумента – y и z, каждый из которых может быть либо переменной, либо результатом некоторой операции. Пусть если оба аргумента являются переменными, то y – переменная с меньшим номером, если оба аргумента являются результатами операций, то y – результат операции с меньшим номером, если один из аргументов – переменная, а другой – результат операции, то переменная – это y. Обозначим $r_m = n + 2p_1 + 4(m - p_1) - 3$. Вычисление значения операции f_m моделируется рядами $r_m, r_m + 1, r_m + 2$ и $r_m + 3$. Обозначим для $j \in \{0, 1, 2, 3\}, k \in \{0, 1, U\}$:

$$y(j,k) = \begin{cases} v(j + r_m, t, k), & \text{при } y = x_t,\ t \in \{1, \ldots, n\}, \\ q(j + r_m, t, k), & \text{при } y = f_t(\cdot),\ t \in \{1, \ldots, p\}; \end{cases}$$

$$z(j,k) = \begin{cases} v(j + r_m, t, k), & \text{при } z = x_t,\ t \in \{1, \ldots, n\}, \\ q(j + r_m, t, k), & \text{при } z = f_t(\cdot),\ t \in \{1, \ldots, p\}. \end{cases}$$

Если $y \in \{x_i\}$, то $y(j, U)$ не определено. Если $z \in \{x_i\}$, то $z(j, U)$ не определено.

Для $k \in \{0, 1, 2, 3\},\ i \in \{1, \ldots, n\},\ j \in \{0, 1\}$
$\quad v(r_m + k, i, j) \notin \{y(k, j), z(k, j)\}, \quad v(r_m + k, i, j).a = v(r_m + k + 1, i, j);$
для $k \in \{0, 1, 2, 3\},\ i \in \{1, \ldots, n\},\ j \in \{0, 1\}$
$\quad v(r_m + k, i, j).b = v(r_m + k + 1, i, j);$
для $k \in \{0, 1, 2, 3\},\ i \in \{1, \ldots, p\},\ j \in \{0, 1, U\}$
$\quad q(r_m + k, i, j) \notin \{y(k, j), z(k, j)\}, \quad q(r_m + k, i, j).a = q(r_m + k + 1, i, j);$
для $k \in \{0, 1, 2, 3\},\ i \in \{1, \ldots, p\},\ j \in \{0, 1, U\}$
$\quad q(r_m + k, i, j).b = q(r_m + k + 1, i, j).$

Другими словами, операция (а соответственно и буква a) не использует и не меняет значения переменных и результатов операций, кроме себя и своих аргументов, а так же, как и буква b, просто переводит фишки в следующий ряд, сохраняя их расположение в ряду.

Пусть $\alpha, \beta \in \{0, 1\}$. Смоделируем вычисление результата операции $f_m(\alpha, \beta)$:

$y(r_m + \alpha + 2\beta, \alpha).a = y(r_m + \alpha + 2\beta + 1, \alpha), y(r_m + \alpha + 2\beta, \neg\alpha).a = undef,$
$z(r_m + \alpha + 2\beta, \beta).a = z(r_m + \alpha + 2\beta + 1, \beta), z(r_m + \alpha + 2\beta, \neg\beta).a = undef,$
$y(r_m + \alpha + 2\beta, U).a = z(r_m + \alpha + 2\beta, U).a = undef,$
$q(r_m + \alpha + 2\beta, m, U).a = q(r_m + \alpha + 2\beta + 1, m, f_m(\alpha, \beta)),$
$q(r_m + \alpha + 2\beta, m, 0).a = q(r_m + \alpha + 2\beta, m, 1).a = undef.$

Если аргументы операции f_m равны α и β, то вычисление значения f_m производится на $(r_m + \alpha + 2\beta)$-м шаге. Если результат вычисления равен 0, то после вычисления фишка находится на состоянии $q(r_m + 4, m, 0)$. Если результат вычисления равен 1, то после вычисления фишка находится на состоянии $q(r_m+4, m, 1)$. Если вычисление значения f_m не было произведено, то фишка находится на состоянии $q(r_m + 4, m, U)$. В примере (см. рисунок) вычисление результата операции $\vee$ производится в 9–12-м рядах, вычисление результата операции $\wedge$ производится в 13–16-м рядах.

Переходы, соответствующие проверке вычисленных значений

Определим действие букв на $(s + 1)$-м и 0-м рядах:

$$\text{для } i \in \{1, \ldots, n\}, \quad v(s+1, i, 1).a = end, \quad v(s+1, i, 0).a = \begin{cases} end, & Q_i = \exists, \\ undef, & Q_i = \forall; \end{cases}$$

$$\text{для } i \in \{1, \ldots, n\}, \quad v(s+1, i, 0).b = v(0, i, 0), \quad v(s+1, i, 1).b = v(0, i, 1).$$

Вычисление истинности формулы F можно будет завершить (применение буквы a в $(s+1)$-м ряду), только если значения всех переменных всеобщности будут равны 1, т. е. значение счетчика станет равным $2^\lambda - 1$. Иначе допустимо лишь применение буквы b, которая переводит все фишки в 0-й ряд, не изменяя значение счетчика.

Для $i \in \{1, \ldots, p-1\}$

$$q(s+1, i, 0).a = q(s+1, i, 1).a = q(s+1, i, U).a = end,$$
$$q(s+1, p, 0).a = q(s+1, p, U).a = undef, q(s+1, p, 1).a = end.$$

Вычисление истинности формулы F можно будет завершить (применение буквы a в $(s + 1)$-м ряду), только если результат p-й операции, а значит и всей функции $P(x_1, \ldots, x_n)$, равен 1. Иначе придется применять букву b.

Для $i \in \{1, \ldots, p-1\}$,

$$q(s+1, i, 0).b = q(s+1, i, 1).b = q(s+1, i, U).b = q(0, i, S),$$
$$q(s+1, p, 0).b = q(s+1, p, U).b = q(0, p, S), q(s+1, p, 1).b = q(0, p, N).$$

Если результат p-й операции равен 0 или неопределен, то при переходе на следующий шаг вычислений фишка, находящаяся в колонке p-й операции, попадет в состояние $q(1, i, S)$ и на следующем шаге значения переменных всеобщности не смогут поменяться. Впоследствии мы докажем, что при действии кратчайшего б.с.с. такого не произойдет. Если результаты всех операций правильно посчитаны, то буква b переводит все фишки в состояния

$q(1,i,S)$ и значение счетчика C на следующем шаге сможет изменяться.

$$\begin{array}{ll}
\text{Для } i\in\{1,\dots,n\},\ k\in\{0,1\} & v(0,i,k).a=v(1,i,k);\\
\text{для } i\in\{1,\dots,p\},\ k\in\{N,S\} & q(0,i,k).a=q(1,i,k);\\
\text{для } i\in\{1,\dots,n\},\ k\in\{0,1\} & v(0,i,k).b=v(0,i,0);\\
\text{для } i\in\{1,\dots,p\},\ k\in\{N,S\} & q(0,i,k).b=q(0,i,S).
\end{array}$$

Буква a просто переводит все фишки в 1-й ряд. Буква b оставляет все фишки в 0-м ряду. При этом фишки в колонках, соответствующих переменным, переводятся в состояния $v(0,i,0)$ или в $q(0,i,S)$. В примере (см. рисунок) $s+1=17$.

Таким образом, описание действия функции переходов автомата $\mathscr{A}$ завершено.

4.5. Доказательство корректности сведения

Докажем, что синхронизация автомата $\mathscr{A}$ соответствует описанному выше алгоритму проверки истинности формулы F.

Пусть автомат $\mathscr{A}$ бережно синхронизируем. Рассмотрим произвольное кратчайшее б.с.с. w для него. Покажем, что w можно представить в виде произведения двух слов: u', соответствующего подготовке и обнулению счетчика C, и u, соответствующего остальной работе алгоритма $SAT(1)$.

Подготовка и обнуление счетчика C (слово u')

Лемма 6. *Справедливо равенство* $w[1,s+1]=b^{s+1}$.

Доказательство. Буква a определена не на всем множестве Q, а буква b на всем, поэтому $w[1]=b$. Пусть $t\in\{1,\dots,s\}$ и $w[1,t]=b^t$. Докажем, что $w[t+1]=b$. Рассмотрим состояние $v(s+1,1,0)$. Буква a не определена на состоянии $v(s+1,1,0)$. Кроме того, $v(s+1-t,1,0).b^t=v(s+1,1,0)$. Состояние $v(s+1-t,1,0)$ существует, так как $t\leqslant s$. Это означает, что $v(s+1,1,0)\in Q.b^t$. Поэтому буква a определена не на всем множестве $Q.b^t$, а значит, $w[t+1]=b$. Следовательно, $w[1,s+1]=b^{s+1}$ и лемма доказана.

Заметим, что

$$Q.b^{s+1}=\big\{v(0,i,j)\mid i\in\{1,\dots,n\},\ j\in\{0,1\}\big\}\cup \\ \cup\big\{q(0,i,S)\mid i\in\{1,\dots,p\}\big\}\cup\big\{q(0,p,N)\big\}\cup\big\{end\big\},$$

т. е. слово b^{s+1} переводит все состояния, кроме состояния end, в 0-й ряд.

Пусть $i \in \{1, \ldots, n\}$, обозначим

$$K_i = \big\{v(m,i,j) \mid m \in \{0,\ldots,s+1\},\ j \in \{0,1\}\big\}.$$

Пусть $i \in \{1, \ldots, p\}$, обозначим

$$K_{n+i} = \big\{q(m,i,j) \mid m \in \{0,\ldots,s+1\},\ j \in \{0,1,U,S,N\},\ q(m,i,j) \in Q\big\}.$$

Таким образом, для $i \in \{1, \ldots, n+p\}$ K_i – это i-я *колонка* состояний. Пусть $m \in \{0, \ldots, s+1\}$, обозначим m-й *ряд* состояний через R_m, т. е.

$$R_m = \big\{v(m,i,j) \mid i \in \{1,\ldots,n\},\ j \in \{0,1\}\big\} \cup \\ \cup \big\{q(m,i,j) \mid i \in \{1,\ldots,p\},\ j \in \{0,1,U,N,S\},\ q(m,i,j) \in Q\big\}.$$

Лемма 7. 1. *Пусть $m \in \{1, \ldots, s\}$, $i \in \{1, \ldots, n+p\}$, $c \in \{a, b\}$, $q \in R_m \cap K_i$ и буква c определена на состоянии q. Тогда состояние $q.c \in R_{m+1} \cap K_i$.*

2. *Пусть $i \in \{1, \ldots, n+p\}$, $q \in R_{s+1} \cap K_i$. Тогда состояние $q.b \in R_0 \cap K_i$, и если буква a определена на состоянии q, то $q.a = end$.*

3. *Пусть $i \in \{1, \ldots, n+p\}$, $q \in R_0 \cap K_i$. Тогда состояние $q.b \in R_0 \cap K_i$, а состояние $q.a \in R_1 \cap K_i$.*

Доказательство. Лемма доказывается простой проверкой по определению функции переходов автомата $\mathscr{A}$.

Другими словами, в большинстве случаев (кроме действия буквы a в $(s+1)$-м ряду и буквы b в 0-м ряду) любая фишка под действием любой буквы переходит в следующий ряд, но остается в прежней колонке. Для каждого $t \in \{1, |w|\}$ обозначим множество $\delta(Q \backslash \{end\}, w[1,t])$ через $\Pi(t)$. Опишем некоторые свойства множеств $\Pi(t)$.

Лемма 8. *Для любого $t \in \{s+1, |w|-1\}$ множество $\Pi(t)$ содержится внутри одного ряда, причем $\Pi(t) \cap K_i \neq \varnothing$ для всех $i \in \{1, \ldots, n+p\}$. Кроме того, $|\Pi(t-1) \cap K_i| \geqslant |\Pi(t) \cap K_i|$.*

Доказательство. В силу леммы 7 легко проверить, что множество $\Pi(s+1)$ целиком содержится в 0-м ряду, причем для $i \in \{1, \ldots, n\}$

$$\Pi(s+1) \cap K_i = \{v(0,i,0), v(0,i,1)\} \neq \varnothing$$

и для $i \in \{n+1, \ldots, n+p-1\}$

$$\Pi(s+1) \cap K_i = \{q(0,i,S)\} \neq \varnothing, \quad \Pi(s+1) \cap K_p = \{q(0,p,S), q(0,p,N)\} \neq \varnothing.$$

Это означает, что для $t = s+1$ лемма доказана.

Пусть $t \in \{s+2, |w|-1\}$, $\Pi(t-1) \subseteq R_m$ для некоторого $m \in \{0, \ldots, s+1\}$ и для всех $i \in \{1, \ldots, n+p\}$, $\Pi(t-1) \cap K_i \neq \varnothing$.

Пусть $m \in \{1, \ldots, s\}$. Несложно заметить, что $\Pi(t) \subseteq R_{m+1}$. Из леммы 7 получаем, что если состояние $q \in \Pi(t-1) \cap K_i$ для некоторого $i \in \{1, \ldots, n\}$, то $q.w[t] \in K_i$. С другой стороны, $q.w[t] \in \Pi(t-1).w[t] = \Pi(t)$. Следовательно, $q.w[t] \in K_i \cap \Pi(t)$, а это значит, что $K_i \cap \Pi(t) \neq \varnothing$.

Пусть $m = 0$. Если $w[t] = b$, то множество $\Pi(t) \subseteq R_0$ и для $i \in \{1, \ldots, n\}$

$$K_i \cap \Pi(t) = \{v(0, i, 0)\} \neq \varnothing,$$

для $i \in \{n+1, \ldots, n+p\}$

$$K_i \cap \Pi(t) = K_i \cap \Pi(t-1) \neq \varnothing.$$

Если $w[t] = a$, то $\Pi(t) \subseteq R_1$ и для любого $i \in \{1, \ldots, n+p\}$ существует состояние $q \in K_i \cap \Pi(t-1)$, которое переходит в состояние $q.w[t] \in K_i \cap \Pi(t)$. Следовательно, $K_i \cap \Pi(t) \neq \varnothing$.

Пусть $m = s+1$. Если $w[t] = b$, то $\Pi(t) \subseteq R_0$ и для любого $i \in \{1, \ldots, n+p\}$ существует состояние $q \in K_i \cap \Pi(t-1)$ такое, что $q.b \in R_0 \cap K_i$. Если $w[t] = a$, то буква a определена на множестве $\Pi(t-1)$ и так как $\Pi(t-1)$ содержится в $(s+1)$-м ряду, то $\Pi(t-1).a = \{end\}$. В этом случае слово $w[1, t]$ является б.с.с. для автомата $\mathscr{A}$. Но $|w[1, t]| < |w|$, что противоречит тому, что слово w – кратчайшее б.с.с. для $\mathscr{A}$.

Для $t \in \{s+2, |w|-1\}$ $|\Pi(t-1) \cap K_i| \geqslant |\Pi(t) \cap K_i|$, так как для любого $i \in \{1, \ldots, n+p\}$, прообразами состояний колонки K_i являются только состояния колонки K_i, поэтому количество фишек в колонке K_i не может увеличиваться. Лемма доказана.

Если для некоторого t, множество $\Pi(t) \subseteq R_m$ для некоторого m, то мы будем говорить, что под действием буквы $w[t]$ *фишки проходят* m-й *ряд*. Из леммы 8 понятно, что для $t \in \{s+2, |w|-1\}$ фишки всегда проходят какой-то ряд.

Благодаря лемме 8 можно корректно определить понятие состояния колонки. Пусть $t \in \{s+1, \ldots, |w|-1\}$, для любого $i \in \{1, \ldots, n+p\}$ $K_i \cap \Pi(t) \neq \varnothing$ и для некоторого $m \in \{0, s+1\}$, $\Pi(t) \subseteq R_m$. *Состояние колонки* после применения слова $w[1, t]$ определяется множеством $K_i \cap \Pi(t)$. Обозначим его через $\pi_i(t)$.

Пусть $i \in \{1, \ldots, n\}$. Если множество $K_i \cap \Pi(t) = \{v(m, i, 0)\}$, то будем говорить, что колонка K_i находится в состоянии 0 и записывать $\pi_i(t) = 0$. Если $K_i \cap \Pi(t) = \{v(m, i, 1)\}$, то $\pi_i(t) = 1$. Если $K_i \cap \Pi(t) = \{v(m, i, 0), v(m, i, 1)\}$, то $\pi_i(t) = 01$.

Пусть $i \in \{n+1, \ldots, n+p\}$. Если $K_i \cap \Pi(t) = \{q(m,i,0)\}$, то $\pi_i(t) = 0$. Если $K_i \cap \Pi(t) = \{q(m,i,1)\}$, то $\pi_i(t) = 1$. Если $K_i \cap \Pi(t) = \{q(m,i,U)\}$, то $\pi_i(t) = U$. Если $K_i \cap \Pi(t) = \{q(m,i,S)\}$, то $\pi_i(t) = S$. Если $K_p \cap \Pi(t) = \{q(m,p,N)\}$, то $\pi_p(t) = N$. Если $K_p \cap \Pi(t) = \{q(m,p,N), q(m,p,S)\}$, то $\pi_p(t) = NS$.

Докажем, что после применения слова $w[1, s+1]$ и до конца синхронизации колонки не могут находиться ни в каких состояниях, кроме определенных выше.

Лемма 9. *Пусть* $t \in \{s+1, |w|-1\}$. *Если* $i \in \{1, \ldots, n\}$, *то* $\pi_i(t) \in \{0, 1, 01\}$. *Если* $i \in \{n+1, \ldots, n+p\}$, *то* $\pi_i(t) \in \{0, 1, U, N, S, NS\}$.

Доказательство. Пусть $t \in \{s+2, |w|-1\}$ и $\Pi(t) \subseteq R_m$.

Следующие импликации получаются непосредственно из определения функции переходов автомата $\mathscr{A}$.

Пусть $i \in \{1, \ldots, n\}$.

Если $\pi_i(t-1) \in \{0,1\}$, то $\pi_i(t) \in \{0,1\}$.

Если $\pi_i(t-1) = 01$ и $m \neq 0$, то $\pi_i(t) = 01$.

Если $\pi_i(t-1) = 01$ и $m = 0$, то $\pi_i(t) \in \{0, 01\}$.

Пусть $i \in \{n+1, \ldots, n+p\}$.

Если $\pi_i(t-1) \in \{0, 1, U, N, S\}$, то $\pi_i(t) \in \{0, 1, U, N, S\}$, так как $|\Pi(t-1) \cap K_i| \geqslant |\Pi(t) \cap K_i| > 0$.

Если $\pi_p(t-1) = NS$ и $m \notin \{0, n\}$, то $\pi_p(t) = NS$.

Если $\pi_p(t-1) = NS$ и $m = n$, то $\pi_p(i) = U$.

Если $\pi_p(t-1) = NS$ и $m = 0$, то $\pi_p(t) \in \{S, NS\}$.

Лемма доказана.

Лемма 10. *Справедливы равенства* $w[1, s+2] = b^{s+2}$, $w[s+3] = a$.

Доказательство. Из леммы 6 имеем $w[1, s+1] = b^{s+1}$. Рассуждая от противного, предположим, что $w[s+2] = a$. Рассмотрим колонку $K_{h(1)}$ ($h(1)$ – номер первой переменной всеобщности). Имеем $\pi_{h(1)}(s+2) = 01$. В состояние *end* могут переводиться только состояния $(s+1)$-го ряда, поэтому $\Pi(|w|-1) \subseteq R_{s+1}$. Но для перехода в состояние *end* колонка $K_{h(1)}$ должна находиться в состоянии 1. Из определения функции переходов автомата видно, что перейти из состояния 01 в другое состояние колонка $K_{h(1)}$ может только в 0-м ряду под действием буквы b, т. е. найдется такое $k \in \{s+3, |w|-1\}$, что $\Pi(k-1) \subseteq R_0$ и $w[k] = b$. Рассмотрим множество $\Pi(k)$. Имеем $\pi_i(k) = 0$ при $i \in \{1, \ldots, n\}$ и $\pi_i(k) = S$ при $i \in \{n+1, \ldots, n+p\}$. Поэтому $Q.w[1,k] = Q.b^{s+2}$ и слово $b^{s+2}w[k+1, |w|]$ также будет б.с.с. для автомата $\mathscr{A}$, что противоречит минимальности длины w. Поэтому $w[s+2] = b$.

Имеем равенства $\pi_i(s+2) = 0$ при $i \in \{1, \ldots, n\}$ и $\pi_i(s+2) = S$ при $i \in \{n+1, \ldots, n+p\}$. В этом случае $\Pi(s+2).b = \Pi(s+2)$. Следовательно, $w[s+3]$ не может быть буквой b (иначе слово $w[1, s+2]w[s+4, |w|]$ также будет б.с.с. для автомата $\mathscr{A}$, что противоречит минимальности длины w). Лемма доказана.

Лемма 11. *Если* $t \in \{s+2, |w|-1\}$, $i \in \{1, \ldots, n+p\}$, *то* $\pi_i(t) \in \{0, 1, U, N, S\}$.

Доказательство. Из леммы 8 следует, что $|\Pi(t-1) \cap K_i| \geqslant |\Pi(t) \cap K_i| > 0$ для всех $i \in \{1, \ldots, n+p\}$ и для всех $t \in \{s+2, |w|-1\}$. После применения слова $w[s+2]$ в каждой колонке останется по одному состоянию. Поэтому $|\Pi(t) \cap K_i| = 1$ для всех $i \in \{1, \ldots, n+p\}$ и для всех $t \in \{s+2, |w|-1\}$. Лемма доказана.

Другими словами, после применения слова $w[1, s+2]$ и до конца синхронизации в каждой колонке фишка стоит ровно на одном состоянии.

Напомним, что алгоритм $SAT(1)$ можно описать, используя счетчик C, значениями которого могут быть числа от 0 до $2^\lambda - 1$. Пусть, как и раньше, $Q_{h(1)}, \ldots, Q_{h(\lambda)}$ – все кванторы всеобщности в формуле F. Значения переменных $x_{h(1)}, \ldots, x_{h(\lambda)}$ составляют двоичную запись значения счетчика C.

Счетчик C также можно определить в терминах автомата $\mathscr{A}$. Пусть $t \in \{s+2, |w|-1\}$, рассмотрим значения $\pi_{h(1)}(t), \ldots, \pi_{h(\lambda)}(t)$. По лемме 11 и определению автомата $\pi_{h(i)}(t) \in \{0, 1\}$ при $i \in \{1, \ldots, \lambda\}$. Определим значение счетчика C как целое число, двоичной записью которого являются записанные по порядку числа $\pi_{h(1)}(t), \ldots, \pi_{h(\lambda)}(t)$. Младшему биту будет соответствовать число $\pi_{h(\lambda)}(t)$. Значение счетчика после применения слова $w[1, t]$, равное $(\pi_{h(1)}(t), \ldots, \pi_{h(\lambda)}(t))$, будем обозначать через $C(t)$.

Напомним, что мы собирались разбить слово w на две части – u' и u. Теперь ясно, что в качестве слова u', моделирующего подготовку и обнуление счетчика в алгоритме $SAT(1)$, можно взять слово $w[1, s+2] = b^{s+2}$.

Описание слова u, моделирующего алгоритм $SAT(1)$

Пусть автомат $\mathscr{A}$ по-прежнему является бережно синхронизируемым и w – кратчайшее б.с.с. для $\mathscr{A}$.

Покажем, что слово $u = w[s+2, |w|\}$ можно представить в виде произведения 2^λ слов $u_0, \ldots, u_{2^\lambda - 1}$, каждое из которых имеет длину $s+2$. При этом для любого $M \in \{0, 2^\lambda - 1\}$ множество $Q.u'u_0 \ldots u_M$ будет содержаться в 0-м ряду. Кроме того, после применения слова $u'u_0 \ldots u_M$ значение счетчика C будет равным M, т. е. $C(|u'| + |u_0| + \ldots + |u_M|) = M$. Применение слова u_M для $M \in \{0, 2^\lambda - 1\}$ будет моделировать M-й шаг алгоритма $SAT(1)$. Применение слова u_M к множеству $Q.u'u_0 \ldots u_{M-1}$ мы будем называть M-м *шагом бережной синхронизации* автомата $\mathscr{A}$.

Лемма 12. *Справедливо равенство*

$$C(s+2) = C((s+2)+n+1) = C((s+2)+s+2) = 0.$$

Доказательство. Из леммы 10 следует, что

$$\pi_{h(1)}(s+2) = \ldots = \pi_{h(\lambda)}(s+2) = 0 \quad \text{и} \quad \pi_{h(1)}(s+3) = \ldots = \pi_{h(\lambda)}(s+3) = 0.$$

Кроме того, $\pi_{n+p}(s+3) = S$, поэтому буква a не определена на множествах $\Pi((s+2)+1), \ldots, \Pi((s+2)+\lambda)$. Следовательно, $\pi_{h(i)}((s+2)+\lambda+1) = \ldots = 0$ для любого $i \in \{1, \ldots, \lambda\}$. Для каждого $i \in \{1, \ldots, \lambda\}$ состояние колонки $K_{h(i)}$ не может измениться под действием букв $w[(s+2)+\lambda+1], \ldots, w[(s+2)+s+1]$. Поэтому $\pi_{h(i)}((s+2)+n+1) = \pi_{h(1)}((s+2)+s+2) = 0$. Тем самым лемма доказана.

Таким образом, в процессе применения слова $w[s+2, 2(s+2)]$ значение счетчика C остается равным 0. Тем самым установлено соответствие между действием слова $w[s+2, 2(s+2)]$ и нулевым шагом алгоритма $SAT(1)$. Докажем теперь, что на M-м шаге синхронизации значение счетчика C не может увеличиться больше чем на 1.

Лемма 13. *Пусть* $k \in \{(s+2)+1, |w|-1\}$, $\Pi(k)$ *содержится в* 1*-м ряду и* $C(k) = M-1$, *тогда*

1. $C(k+n) = C(k+s) \in \{M, M-1\}$;

2. *если* $C(k+n) = M$ *и* τ *– номер младшего единичного бита числа* $M-1$, *то* $C(k+\sigma^{-1}(h(\tau))-1) = M-1$ *и* $C(k+\sigma^{-1}(h(\tau))) = M$.

Доказательство. Из определения функции переходов автомата $\mathscr{A}$ следует, что на M-м шаге синхронизации состояния колонок переменных всеобщности $K_{h(1)}, \ldots, K_{h(\lambda)}$ могут изменяться только под действием тех букв $w[t]$, при которых множество $\Pi(t)$ содержится в одном из рядов $R_1, \ldots, R_\lambda$. Поэтому $C(k+\lambda) = C(k+n) = C(k+s)$. Состояния колонок $K_{h(1)}, \ldots, K_{h(\lambda)}$ могут измениться только под действием буквы a. Следовательно, если слово $w[k+1, k+\lambda] = b^\lambda$, то значение счетчика на M-м шаге не изменится.

Буква $w[t]$ может быть равной a, только если колонка K_{n+p} находится в состоянии N. После применения буквы a колонка K_{n+p} переходит в состояние S. Поэтому в слове $w[k+1, k+\lambda]$ не может быть более одной буквы a. Следовательно, значение счетчика не может измениться более одного раза.

Допустим, что $w[t]$ – единственная буква a в слове $w[k+1, k+\lambda]$. Из определения функции переходов автомата $\mathscr{A}$ следует, что значения счетчика до и после действия буквы a имеют вид

$$C(t-1) = (*, 0, 1, \ldots, 1), \; C(t) = (*, 1, 0, \ldots, 0),$$

где звездочками обозначены одинаковые цепочки битов, длину которых обозначим $\tau - 1$. Поскольку значение счетчика изменяется один раз, получаем $C(t-1) = M - 1$, а значит, τ – это номер младшего единичного бита числа $M - 1$. Кроме того, сравнивая вид $C(t-1)$ и $C(t)$, заключаем, что $C(t) = M$.

Напомним, что через $\sigma(m)$ обозначается номер переменной, значение которой присваивается в ряду R_m. В частности, $\sigma^{-1}(h(\tau))$ – это номер ряда, в котором моделируется увеличение значения переменной всеобщности с номером τ. По лемме 7 каждая буква сдвигает фишки в следующий ряд, значит, $t = k + \sigma^{-1}(h(\tau))$. Лемма доказана.

Рассмотрим подробнее M-й шаг алгоритма $SAT(1)$. Обозначим через $x_1^M, \ldots, x_n^M$ значения переменных $x_1, \ldots, x_n$ сразу после завершения выполнения M-го шага алгоритма. Откажемся от требования бережной синхронизируемости автомата $\mathscr{A}$. Опишем построение некоторого слова $w \in \Sigma^*$ по алгоритму $SAT(1)$, которое в случае истинности формулы F будет являться б.с.с. для автомата $\mathscr{A}$.

Префикс $w[1, s+2]$ положим равным $u' = b^{s+2}$. Пусть уже построен префикс слова w длины k. Как и раньше, через $\Pi(k)$ мы будем обозначать множество $\delta(Q \backslash \{end\}, w[1,k])$, через $\pi_i(k)$ – состояние колонки k_i после применения слова $w[1,k]$, через $C(k)$ – значение счетчика после применения слова $w[1,k]$.

Лемма 14. *Пусть $k \geqslant 2(s+2)$, префикс $w[1,k]$ определен, множество $\Pi(k)$ содержится в 1-м ряду, $\pi_1(k) = x_1^{M-1}, \ldots, \pi_n(k) = x_n^{M-1}$ для некоторого $M \in \{1, \ldots, 2^\lambda - 1\}$ и $\pi_{n+p}(k) = N$, тогда существует слово w_0^M длины n, которое при воздействии на множество $\Pi(k)$ переводит колонки $K_1, \ldots, K_n$ в состояния $x_1^M, \ldots, x_n^M$ соответственно.*

Доказательство. В начале M-го шага алгоритма $SAT(1)$ значение счетчика равно $M-1$, в конце – M. Следовательно, $C(k) = \left(x_{h(1)}^{M-1}, \ldots, x_{h(\lambda)}^{M-1}\right)$. В силу пункта 2 леммы 13 получаем что состояния колонок $K_{h(1)}, \ldots, K_{h(\lambda)}$ могут измениться только при прохождении фишками ряда с номером $\sigma^{-1}(h(\tau))$, где τ – номер младшего нулевого бита числа $M - 1$. Поэтому префикс $w_0^M[1, \lambda]$ можно взять равным $b^{\lambda-\tau}ab^{\tau-1}$. Под действием слова $b^{\lambda-\tau}ab^{\tau-1}$ колонки $K_{h(1)}, \ldots, K_{h(\lambda)}$ перейдут в состояния $x_{h(1)}^M, \ldots, x_{h(\lambda)}^M$, а остальные колонки из первых n (колонки переменных существования) останутся в состояниях x_i^{M-1}.

На M-м шаге алгоритма могут измениться значения переменных существования с номерами, большими чем $h(\tau)$. После применения слова $b^{\lambda-\tau}ab^{\tau-1}$ колонки $K_{h(\tau+1)}, \ldots, K_{h(\lambda)}$ находятся в состоянии 0, а колонка $K_{h(\tau)}$ находится в состоянии 1. Поэтому для всех $i > \tau$ таких, что $\sigma^{-1}(i) \in \{\lambda + 1, n\}$, при прохождении состояниями рядов с номерами $\sigma^{-1}(i)$ могут быть применены

буквы b и a, а если $i < \tau$ и $\sigma^{-1}(i) \in \{\lambda+1, n\}$, то может быть применена только буква b. Следовательно, $w_0^M[\lambda+1, \lambda+h(\tau)-\tau]$ можно взять равным $b^{h(\tau)-\tau}$, а для $i > \tau$ таких, что $\sigma^{-1}(i) \in \{\lambda+1, n\}$, можно положить

$$w_0^M[\sigma^{-1}(i)] = \begin{cases} a, & \text{если } x_i^{M-1} \neq x_i^M, \\ b, & \text{если } x_i^{M-1} = x_i^M \end{cases}$$

(из определения функции переходов видно, что в $\sigma^{-1}(i)$-м ряду буква a меняет значение переменной x_i, а буква b не меняет). В этом случае после применения слова $w_0^M[\lambda+1, n]$ состояния колонок, соответствующих переменным существования, будут равны значениям переменных после M шага алгоритма $SAT(1)$. Лемма доказана.

Благодаря лемме 14 подслово $w[k+1, k+n]$ можно взять равным w_0^M.

Нулевой шаг бережной синхронизации автомата $\mathscr{A}$ также моделирует нулевой шаг алгоритма $SAT(1)$. Только состояния колонок $K_{h(1)}, \ldots, K_{h(\lambda)}$ равны 0 и не могут изменяться, в то время как состояния колонок соответствующих переменным существования в начале ($k = s+3$) равны 0 и при прохождении рядов с номерами $\lambda+1, \ldots, n$ могут измениться в зависимости от того, какие значения присваиваются переменным существования на 0-м шаге алгоритма $SAT(1)$.

Поэтому точно так же, как в лемме 14, для $M = 0$ строится слово w_0^0 длины n, переводящее колонки $K_1, \ldots, K_n$ из состояний 0 в состояния $x_1^0, \ldots, x_n^0$.

Лемма 15. *Пусть $k \geqslant (s+2)+n$, префикс $w[1,k]$ определен, $\Pi(k)$ содержится в $(n+1)$-м ряду,*

$$\pi_1(k) = x_1^M, \ldots, \pi_n(k) = x_n^M \quad \text{и} \quad \pi_{n+1}(k) = \ldots = \pi_{n+p}(k) = U,$$

тогда существует слово w_1^M длины $s-n$, которое при действии на множество $\Pi(k)$ оставляет колонки $K_1, \ldots, K_n$ в состояниях $x_1^M, \ldots, x_n^M$ и переводит колонку состояния 0 или 1, причем колонка K_{n+p} переводится в состояние, совпадающее со значением $P(x_1, \ldots, x_n)$.

Доказательство. Напомним, что в формуле F всего p операций, среди которых p_1 операций $\neg$, которые выполняются в первую очередь, и только после них выполняются все операции $\wedge$ и $\vee$. Применению каждой операции $\neg$ в слове w_1^M будут соответствовать по 2 буквы, каждой операции $\wedge$ или $\vee$ – по 4 буквы. В $n+1$-м ряду колонки с номерами $n+1, \ldots, n+p$ находятся в состоянии U. После вычисления значения операции соответствующая колонка будет находиться в состоянии 0 или 1, совпадающем с результатом операции.

Пусть $m \in \{1, \ldots, p_1\}$. Операции f_m будет соответствовать подслово $w_1^M[2m-1, 2m]$, при этом фишки будут проходить ряды с номерами $n+2m-1$ и $n+2m$. Пусть операция $f_m = \neg$ применяется к переменной x. Положим

$$w_1^M[2m-1, 2m] = \begin{cases} ab, & \text{если } x = 0, \\ ba, & \text{если } x = 1. \end{cases}$$

Применение буквы a соответствует вычислению значения операции. Другими словами, если $x = 0$, то буква a может быть применена при прохождении фишками $(n + 2m - 1)$-го ряда и значение вычисляется при помощи буквы $w_1^M[2m - 1]$; если $x = 1$, буква a может быть применена при прохождении фишками $(n + 2m)$-го ряда и значение вычисляется при помощи буквы $w_1^M[2m]$. После применения слова $w_1^M[1, 2p_1]$ колонки $K_{n+1}, \ldots, K_{n+p_1}$ окажутся в состояниях 0 или 1, совпадающих с результатами операций $f_1, \ldots, f_{p_1}$.

Пусть $m \in \{p_1 + 1, \ldots, p\}$. Операции f_m будет соответствовать подслово $w_1^M[2p_1+4m-3, 2p_1+4m]$, при этом фишки будут проходить ряды с номерами от $n + 2p_1 + 4m - 3$ до $n + 2p_1 + 4m$. Пусть операция $f_m \in \{\wedge, \vee\}$ применяется к аргументам y и z, каждый из которых является либо переменной, либо результатом выполнения другой операции. Пусть так же, как в определении функции перехода автомата, если y и z оба являются переменными, то y – переменная с меньшим номером; если y и z оба являются результатами операций, то y – результат операции с меньшим номером; если один из y и z – переменная, а другой – результат операции, то переменная – это y. Положим

$$w_1^M[2p_1 + 4m - 3, 2p_1 + 4m] = \begin{cases} abbb, & \text{если } y = 0,\ z = 0, \\ babb, & \text{если } y = 0,\ z = 1, \\ bbab, & \text{если } y = 1,\ z = 0, \\ bbba, & \text{если } y = 1,\ z = 1. \end{cases}$$

Применение буквы a соответствует вычислению значения операции. Если $y = 0$, $z = 0$, то буква a может быть применена в $(n + 2p_1 + 4m - 3)$-м ряду и значение вычисляется при помощи буквы $w_1^M[n + 2p_1 + 4m - 3]$, аналогично и при других комбинациях значений y и z. После применения слова $w_1^M[2p_1 + 1, 2p_1 + 4(p - p_1)]$ колонки $K_{n+p_1+1}, \ldots, K_{n+p}$ окажутся в состояниях 0 или 1, совпадающих с результатами вычислений результатов операций $f_{p_1+1}, \ldots, f_p$. Соответственно колонка K_{n+p} попадет в состояние, совпадающее со значением $P(x_1, \ldots, x_n)$. Лемма доказана.

Благодаря лемме 15 подслово $w[k+n+1, k+s]$ можно взять равным w_1^M. Также можно заметить, что если состояния колонок $K_1, \ldots, K_n$ перед применением слова w_1^M совпадают со значениями переменных $x_1, \ldots, x_n$ после

присвоения значений на M-м шаге алгоритма $SAT(1)$ и после применения слова w_1^M состояние колонки K_{n+p} стало равным α, то значение функции $P(x_1, \ldots, x_n)$, вычисленное на M-м шаге алгоритма $SAT(1)$, получится также равным α. Это значит, что для леммы 10 в некотором смысле верна обратная импликация.

Лемма 16. *Пусть $k \geqslant 2(s+2)$, префикс $w[1,k]$ определен, $\Pi(k)$ содержится в 0-м ряду, $C(k) = M$, тогда*

если $w[k+1] = a$, то $\Pi(k+1) \subseteq R_1$, $C(k+1) = M$;

если $w[k+1] = b$, то $\Pi(k+1) \subseteq R_0$, $C(k+1) = 0$.

Доказательство. Лемма доказывается простой проверкой.

Лемма 16 означает, что при прохождении фишками 0-го ряда значение счетчика C может или остаться прежним (применение буквы a), или стать равным 0 (применение буквы b). Поэтому при синхронизации, в моменты, когда фишки находятся в 0-м ряду, нет смысла применять букву b.

Нам осталось описать поведение состояний колонок при прохождении фишками $(s+1)$-го ряда.

Лемма 17. *Пусть $k \geqslant (s+2)+s+1$, префикс $w[1,k]$ определен, $\Pi(k)$ содержится в $(s+1)$-м ряду. Если $w[k+1] = b$, то*

для $i \in \{1, \ldots, n\}$, $\pi_i(k+1) = \pi_i(k)$;

для $i \in \{n+1, \ldots, n+p-1\}$, $\pi_i(k+1) = S$;

для $i = n+p$, если $\pi_{n+p}(k) = 1$, то $\pi_{n+p}(k+1) = N$;

иначе $\pi_{n+p}(k+1) = S$.

Буква a определена на $\Pi(k)$, только если $C(k) = 2^\lambda - 1$ и $\pi_{n+p}(k) = 1$, в этом случае $w[k+1]$ – последняя буква слова w и слово w – б.с.с. для автомата $\mathscr{A}$.

Доказательство. Лемма доказывается простой проверкой.

Другими словами, лемма 17 утверждает, что бережная синхронизация автомата $\mathscr{A}$ может быть завершена, только если значение счетчика равно $2^\lambda - 1$. В противном случае синхронизация переходит на следующий шаг. Причем колонка K_{n+p} переходит в состояние N только в случае, если результат операции f_p вычислен и равен 1. В противном случае колонка K_{n+p} окажется в состоянии S, и на следующем шаге синхронизации значение счетчика не сможет измениться.

4.6. Случай истинной формулы F

Допустим, что формула $F = Q_1x_1Q_2x_2\dots Q_nx_nP(x_1\dots x_n)$ является истинной. Докажем, что в этом случае ЧКА $\mathscr{A}$ бережно синхронизируем.

Лемма 18. *Пусть формула F истинна. Пусть для $M \in \{0,\dots,2^\lambda-2\}$ слово $u_M = aw_0^Mw_1^Mb$, для $M = 2^\lambda-1$ слово $u_M = aw_0^Mw_1^Ma$ и $u' = b^{s+2}$, тогда слово $w = u'u_0\dots u_{2^\lambda-1}$ будет кратчайшим б.с.с. для автомата $\mathscr{A}$ и $|w| = |u'u_0\dots u_{2^\lambda-1}| = (s+2)(2^\lambda+1)$.*

Доказательство. Нетрудно проверить, что множество $\delta(Q\backslash\{end\}, b^{s+2})$ содержится в 0-м ряду, и после применения слова $u' = b^{s+2}$ значение счетчика C равно 0.

Докажем, что $u'u_0\dots u_{2^\lambda-1}$ является б.с.с. для автомата $\mathscr{A}$. Из леммы 12 и замечания после леммы 14 следует, что множество $\delta(Q\backslash\{end\}, u'aw_0^0)$ содержится в 0-м ряду, значение счетчика C после применения слова $u'u_0$ равно 0 и состояния колонок $K_1,\dots,K_n$ совпадают со значениями переменных $x_1,\dots,x_n$ после выполнения 0-го шага. Из лемм 16, 12 и 15 следует, что после применения слова $u'u_0[1,|u_0|-1] = u'aw_0^0w_1^0$ значения всех операций вычислены и колонки $K_{n+1},\dots,K_{n+p}$ находятся в состояниях 0 или 1. Кроме того, из истинности формулы F следует, что после 0-го шага работы алгоритма $SAT(1)$ результат вычисления функции $P(x_1,\dots,x_n)$ равен 1, следовательно, колонка K_{n+p} находится в состоянии 1. Поэтому после применения слова $u'u_0$ состояние колонки K_{n+p} равно N.

Далее, пусть для некоторого номера $M \in \{1,\dots,2^\lambda-1\}$ множество $\delta(Q\backslash\{end\}, u'u_0\dots u_{M-1})$ содержится в 0-м ряду, значение счетчика C после применения слова $u'u_0\dots u_{M-1}$ равно $M-1$ и колонка K_{n+p} находится в состоянии N. Пусть $\bar{u}^M = u'u_0\dots u_{M-1}aw_0^Mw_1^M$. Из лемм 16, 13 и 14 следует, что после применения слова $\bar{u}^M$ значения счетчика C равно M и состояния колонок $K_1,\dots,K_n$ совпадают со значениями переменных $x_1,\dots,x_n$ после выполнения M-го шага алгоритма $SAT(1)$ (значение счетчика и значения переменных существования изменяются по действием слова w_0^M). Из леммы 15 следует, что после применения слова $\bar{u}^M$ колонки $K_{n+1},\dots,K_{n+p}$ находятся в состояниях, совпадающих с результатами операций $f_1,\dots,f_p$, посчитанными на M-м шаге. Кроме того, из истинности формулы F следует, что после M-го шага работы алгоритма $SAT(1)$ результат вычисления функции $P(x_1,\dots,x_n)$ равен 1, следовательно, колонка K_{n+p} находится в состоянии 1.

Если $M < 2^\lambda-1$, то по лемме 17 после применения слова $u'u_0\dots u_M = \bar{u}^Mb$ состояние колонки K_{n+p} равно N, $\delta(Q\backslash\{end\}, \bar{u}^Mb)$ содержится в 0-м ряду и синхронизация автомата переходит на начало $(M+1)$-го шага. Если $M = 2^\lambda-1$, то по лемме 17 после применения слова $\bar{u}^Ma = u'u_0\dots u_{2^\lambda-1}$

все состояния автомата $\mathscr{A}$ переходят в состояние end и бережная синхронизация заканчивается. Следовательно, слово $u'u_0 \dots u_{2^\lambda-1}$ является б.с.с. для автомата $\mathscr{A}$.

Докажем, что слово $w = u'u_0 \dots u_{2^\lambda-1}$ является кратчайшим б.с.с. Предположим w' – кратчайшее б.с.с. для автомата $\mathscr{A}$. Докажем, что его длина $|w'| \geqslant (s+2)(2^\lambda+1)$. Из леммы 10 следует, что $w'[1, s+2] = b^{s+2}$ и после применения слова b^{s+2} значение счетчика равно 0. Из леммы 8 получаем, что для всех $t \in \{s+2, \dots, |w'|-1\}$ множество $\delta(Q\backslash\{end\}, w'[1,t])$ содержится в одном ряду и для всех $i \in \{1, \dots, n+p\}$ $\delta(Q\backslash\{end\}, w'[1,t]) \cap K_i \neq \varnothing$. Следовательно, бережная синхронизация может завершиться после применения буквы a в $(s+1)$-м ряду. Если буква a определена на подмножестве $(s+1)$-го ряда, то значение счетчика при этом должно быть равно $2^\lambda - 1$, так как буква a в колонках $K_{h(1)}, \dots, K_{h(\lambda)}$ ряда R_{s+1} определена только на состояниях $v(s+1, h(1), 1), \dots, v(s+1, h(\lambda), 1)$. Пусть $w' = u'v_0v_1 \dots v_\mu$, где для всех $t \in \{0, \mu-1\}$ $\delta(Q\backslash\{end\}, u'v_0 \dots v_t) \subseteq R_0$ и $\delta(Q\backslash\{end\}, u'v_0 \dots v_tv_{t+1}[1]) \subseteq R_1$. Тогда из лемм 16, 12 и 15 следует, что после применения слова $u'v_0$ значение счетчика равно 0. Пусть для некоторого $M \in \{1, \dots, \mu\}$ после применения слова $u'v_0 \dots v_{M-1}$ значение счетчика равно $M_0 - 1$. Из леммы 12 имеем, что после применения слова $u'v_0 \dots v_{M-1}v_M[1, s+1]$ значение счетчика равно $M_0 - 1$ или M_0. Если получившееся значение счетчика меньше $2^\lambda - 1$, то по лемме 17 $v_M[s+2] = b$ и значение счетчика не изменяется. Если следующая буква слова w это b, то она принадлежит уже слову v_{M+1}. Если же длина слова v_M больше, чем $s+2$, то слово $v_M[s+3, |v_M|]$ состоит из букв a и после его применения значение счетчика становится равным 0. Следовательно, после применения слова $u'v_0 \dots v_M$ значение счетчика равно 0, $M_0 - 1$ или M_0, т. е. не превосходит M_0.

Таким образом, значение счетчика может стать равным $2^\lambda - 1$ не быстрее чем за 2^λ применений слов типа v_M. Поэтому $\mu \geqslant 2^\lambda - 1$. Для всех $M \in \{0, \dots, \mu\}$ $|v_M| \geqslant s+2$. Следовательно, длина слова

$$|w'| = |u'| + |u_0| + \dots + |u_\mu| \geqslant (s+2) + 2^\lambda(s+2) = (2^\lambda+1)(s+2)$$

и слово $w = u'u_0 \dots u_{2^\lambda-1}$ является кратчайшим б.с.с. для автомата $\mathscr{A}$. Лемма доказана.

Из леммы 18 следует, что если формула F истинна, то построенный в соответствии с ней автомат $\mathscr{A}$ является бережно синхронизируемым.

4.7. Случай ложной формулы F

Докажем, что если ЧКА $\mathscr{A}$ бережно синхронизируем и $w \in \Sigma^*$ – кратчайшее б.с.с. для $\mathscr{A}$, то по слову w можно построить реализацию алгоритма

$SAT(1)$. Иными словами, если w – кратчайшее б.с.с. для $\mathscr{A}$, то оракул в алгоритме $SAT(1)$ может делать выбор значений переменных в соответствии со словом w, и результат каждого вычисления значения функции $P(x_1, \ldots, x_n)$ будет получаться равным 1. Для начала опишем структуру любого кратчайшего б.с.с. для автомата $\mathscr{A}$. Заметим, что пока мы не предполагаем истинность формулы F.

Лемма 19. *Пусть слово w есть кратчайшее б.с.с. для автомата $\mathscr{A}$. Тогда $w = u'u_0 \ldots u_{2^\lambda-1}$, где $u' = b^{s+2}$, $|u_0| = \ldots = |u_{2^\lambda-1}| = s+2$ и для всех $M \in \{0, \ldots, 2^\lambda - 2\}$ множество $\delta(Q\backslash\{end\}, u'u_0 \ldots u_M)$ лежит в 0-м ряду, для этого множества значение счетчика C равно M.*

Доказательство. Из леммы 10 получаем, что $w[1, s+2] = b^{s+2} = u'$. Пусть так же, как в доказательстве леммы 18, $w = u'v_0v_1 \ldots v_\mu$, где для $t \in \{0, \mu - 1\}$, $\delta(Q\backslash\{end\}, u'v_0 \ldots v_t) \subseteq R_0$ и $v_{t+1}[1]$ не меняет значение счетчика, т. е. $v_{t+1}[1] = a$. Рассуждая аналогично второй половине доказательства леммы 18, несложно получить, что $\mu \geqslant 2^\lambda - 1$ (благодаря леммам 12–17 можно показать, что значение счетчика может увеличиться до $2^\lambda - 1$, не быстрее чем за 2^λ шагов синхронизации). Докажем, что $\mu = 2^\lambda - 1$.

Рассуждая аналогично доказательству леммы 18, можно заметить, что каждое слово v_M для $M \in \{1, \ldots, 2^\lambda - 1\}$ может увеличить значение счетчика C на 1, оставить значение счетчика прежним или обнулить его. Предположим от противного, что $\mu > 2^\lambda - 1$. В этом случае найдется такое $M \in \{1, \ldots, 2^\lambda - 1\}$, что слово v_M не увеличивает значение счетчика C. Возьмем минимальное M с таким свойством. Перед применением слова v_M значение счетчика равно $M - 1$. Значение счетчика может уменьшиться только при применении буквы b при прохождении фишками 0-го ряда, в этом случае $|v_M| > s+2$. Поэтому если $|v_M| = s+2$, то после применения слова v_M значение счетчика осталось равным $M - 1$, если $|v_M| > s+2$, то после применения слова v_M значение счетчика стало равным 0.

Пусть $|v_M| = s + 2$. Рассмотрим слово

$$u'v_0 \ldots v_{M-1}[1, \lambda]v_{(M-1)\oplus M}v_M[n+1, s+2]v_{M+1} \ldots v_\mu,$$

где слово $v_{(M-1)\oplus M}$ имеет длину $n - \lambda$ и для каждого $t \in \{1, \ldots, n - \lambda\}$ либо $v_{(M-1)\oplus M}[t] = a$ при $v_{M-1}[\lambda + t] \neq v_M[\lambda + t]$, либо $v_{(M-1)\oplus M}[t] = b$ при $v_{M-1}[\lambda + t] = v_M[\lambda + t]$. Значение счетчика увеличивается при прохождении одного из рядов $R_1, \ldots, R_\lambda$ (поскольку ровно одна из букв $v_M[1], \ldots, v_M[\lambda]$ равна a) и остается прежним при прохождении остальных рядов, поэтому после действия слова $u'v_0 \ldots v_{M-1}[1, \lambda]$ значение счетчика будет равным $M - 1$. Слово $v_{(M-1)\oplus M}$ изменяет состояния только у колонок с номерами

$\{1,\dots,n\}\backslash\{h(1),\dots,h(\lambda)\}$, т. е. колонок переменных существования. Слова $v_{M-1}[n+1,s+2]$ и $v_M[1,\lambda]$ не изменяют состояние этих колонок. Поэтому из определения слова $v_{(M-1)\oplus M}$ несложно понять, что слово $v_{(M-1)\oplus M}$ изменяет состояния колонок с номерами $\{1,\dots,n\}\backslash\{h(1),\dots,h(\lambda)\}$ на те же значения, что и слово

$$v_{M-1}[\lambda+1,n]v_{M-1}[n+1,s+2]v_M[1,\lambda]v_M[\lambda+1,n] = v_{M-1}[\lambda+1,s+2]v_M[1,n].$$

Таким образом, под действием слова $u'v_0\dots v_{M-1}[1,\lambda]v_{(M-1)\oplus M}$ колонки $K_1,\dots,K_n$ (счетчик и переменные существования) переводятся в те же состояния, что и под действием слова $u'v_0\dots v_{M-1}v_M[1,n]$. Также под действием обоих слов колонки $K_{n+1},\dots,K_{n+p}$ переводятся в состояние U. Следовательно, $Q.u'v_0\dots v_{M-1}[1,\lambda]v_{(M-1)\oplus M} = Q.u'v_0\dots v_{M-1}v_M[1,n]$, а значит, и

$$Q.u'v_0\dots v_{M-1}[1,\lambda]v_{(M-1)\oplus M}v_M[n+1,s+2]v_{M+1}\dots v_\mu = Q.w,$$

что противоречит тому, что слово w является кратчайшим. В итоге при $|v_M| = s+2$ мы пришли к противоречию.

Пусть $|v_M| > s+2$. Это означает, что в 0-м ряду была применена буква b. После этого для всех $i \in \{1,\dots,n\}$ состояние колонок K_i равно 0, для всех $i \in \{n+1,\dots,n+p\}$ состояние колонок K_i равно S. В этом случае $Q.u'v_0\dots,v_Mb = Q.b^{s+2}$ и слово $b^{s+2}v_{M+1}\dots v_\mu$ будет б.с.с. для $\mathscr{A}$, что противоречит тому, что слово w является кратчайшим.

Тем самым мы доказали, что $\mu = 2^\lambda - 1$. Повторив только что описанные для $|v_M| > s+2$ рассуждения, можно показать для всех $M \in \{1,\dots,\mu\}$ длина $|v_M| = s+2$. Поэтому для всех $M \in \{1,\dots,2^\lambda-1\}$ можно положить $u_M = v_M$. В этом случае множество $\delta(Q\backslash\{end\}, u'u_0\dots u_M)$ лежит в 0-м ряду, и для этого множества значение счетчика равно M. Лемма доказана.

Пусть теперь формула $F = Q_1x_1Q_2x_2\dots Q_nx_nP(x_1\dots x_n)$ является ложной. В этом случае найдется такое значение счетчика, при котором алгоритм $SAT(1)$, определив значения всех переменных $x_1,\dots,x_n$ и вычислив значение функции $P(x_1,\dots,x_n)$, получит в результате 0.

Лемма 20. *Если формула $F = Q_1x_1Q_2x_2\dots Q_nx_nP(x_1\dots x_n)$ ложна, то построенный в соответствии с ней автомат $\mathscr{A}$ не является бережно синхронизируемым.*

Доказательство. Пусть формула F ложна. Предположим от противного, что автомат, построенный по формуле F, является бережно синхронизируемым. В этом случае по лемме 19 кратчайшее б.с.с. w для автомата $\mathscr{A}$ имеет вид $w = u'u_0\dots u_{2^\lambda-1}$, где $u' = b^{s+2}$, $|u_0| = \dots = |u_{2^\lambda-1}| = s+2$ и для всех

$M \in \{0, \dots, 2^\lambda - 2\}$ множество $\delta(Q \backslash \{end\}, u' u_0 \dots u_M)$ лежит в 0-м ряду, для этого множества значение счетчика C равно M.

Напомним, что на M-м шаге алгоритма $SAT(1)$ к счетчику C прибавляется 1 и у оракула запрашиваются значения переменных существования с номерами большими, чем $h(\tau)$, где τ – номер младшего ненулевого бита числа M. Других вызовов оракула в алгоритме $SAT(1)$ не происходит. Построим по слову w правило, позволяющее оракулу указывать значения переменных существования так, чтобы $P(x_1, \dots, x_n)$ всегда получалось равным 1.

Будем считать, что перед 0-м шагом алгоритма $SAT(1)$ значения всех переменных $x_1, \dots, x_n$ равны 0. Пусть $M \in \{0, \dots, 2^\lambda - 1\}$, в начале M-го шага значения переменных $x_1, \dots, x_n$ равны $x_1^{M-1}, \dots, x_n^{M-1}$ соответственно, значение счетчика C равно $M-1$, и на всех предыдущих шагах алгоритма вычисленное значение $P(x_1, \dots, x_n)$ было равным 1. Докажем, что на M-м шаге переменным $x_1, \dots, x_n$ могут быть присвоены такие значения $x_1^M, \dots, x_n^M$, что $P(x_1^M, \dots, x_n^M)$ будет равным 1.

На M-м шаге алгоритма оракул будет отвечать на запросы в соответствии со значениями $x_1^{M-1}, \dots, x_n^{M-1}$ и словом $u_M[\lambda+2, n+1]$ (под действием этого слова фишки в автомате проходят ряды с номерами $\lambda+1, \cdots, n$). Пусть x_j – переменная существования, тогда оракул присвоит ей следующее значение:

$$x_j^M = \begin{cases} x_j^{M-1}, & u_M[1+\lambda+\sigma^{-1}(j)] = b, \\ \neg\, x_j^{M-1}, & u_M[1+\lambda+\sigma^{-1}(j)] = a. \end{cases}$$

Другими словами, если в $\sigma^{-1}(j)$-м ряду (в ряду, в котором присваивается значение переменной x_j) была применена буква b, то значение переменной x_j на M-м шаге не изменится, а если буква a – то изменится.

Из свойств w несложно понять, что под действием слова $u_M[n+2, s+1]$ колонка K_{n+p} переходит в состояние 1. Поэтому, исходя из замечания, высказанного после леммы 15, получаем, что $P(x_1^M, \dots, x_n^M) = 1$ (действие слова $u_M[n+2, s+1]$ полностью соответствует вычислению значения $P(x_1^M, \dots, x_n^M)$). Тем самым мы получили требуемое.

Мы получили, что для всех $M \in \{0, \dots, 2^\lambda - 1\}$, $P(x_1^M, \dots, x_n^M) = 1$. Следовательно, формула F является истинной, что противоречит нашему предположению. Лемма доказана.

Из лемм 18 и 20 следует, что формула F истинна тогда и только тогда, когда построенный в соответствии с ней автомат $\mathscr{A}$ является бережно синхронизируемым. Формула F содержит n переменных и p операций. Автомат $\mathscr{A}$ содержит 2 буквы и $2n(s+2) + (p-1+2)(n+1) + 3p(s+1-n) + 1$ состояние, что не превосходит $2n(4p+n+2) + (p+1)(n+1) + 3p \cdot 4p + 2$ и, следовательно, полиномиально зависит от размера формулы F. Тем самым

мы доказали полиномиальную сводимость задачи 3-ВЫП С КВАНТОРАМИ к задаче 2-БЕР СИНХ. Предложение 3 доказано.

5. Основной результат

Предложение 4. *Задачи* 2-БЕР СИНХ ДЛИНЫ L *и* КРАТ 2-БЕР СИНХ PSPACE-*трудны.*

Доказательство. Сведем к задачам 2-БЕР СИНХ ДЛИНЫ L и КРАТ 2-БЕР СИНХ задачу 3-ВЫП С КВАНТОРАМИ. Пусть на вход задачи 3-ВЫП С КВАНТОРАМИ подается формула

$$F = Q_1 x_1 Q_2 x_2 \dots Q_n x_n P(x_1 \dots x_n).$$

Построим ЧКА $\mathscr{A}$ так же, как в доказательстве предложения 3. Возьмем число L равным $(s+2)(2^\lambda+1)$, где λ – количество кванторов всеобщности в формуле F, а $s = n + 2p_1 + 4(p - p_1)$, где n – количество переменных в формуле F; p – количество операций в формуле F; p_1 – количество операций $\neg$(не) в формуле F.

Из лемм 18 и 20 следует, что формула F истинна тогда и только тогда, когда построенный в соответствии с ней автомат $\mathscr{A}$ является бережно синхронизируемым и существует кратчайшее б.с.с. w для автомата $\mathscr{A}$. Из лемм 18 и 19 следует, что если слово w существует, то$|w| = (s+2)(2^\lambda+1) = L$. Поэтому в качестве входных данных для любой из задач 2-БЕР СИНХ ДЛИНЫ L и КРАТ 2-БЕР СИНХ можно взять автомат $\mathscr{A}$ и число L. Следовательно,

- формула F истинна тогда и только тогда, когда существует слово длины L, бережно синхронизирующее автомат $\mathscr{A}$. Тем самым мы свели задачу 3-ВЫП С КВАНТОРАМИ к задаче 2-БЕР СИНХ ДЛИНЫ $L = (s+2)(2^\lambda+1)$;
- формула F истинна тогда и только тогда, когда существует кратчайшее слово длины $L = (s+2)(2^\lambda+1)$, бережно синхронизирующее автомат $\mathscr{A}$. Тем самым мы свели задачу 3-ВЫП С КВАНТОРАМИ к задаче КРАТ 2-БЕР СИНХ.

Предложение доказано.

Теорема 1. 1. *Задачи* БЕР СИНХ, БЕР СИНХ ДЛИНЫ L *и* КРАТ БЕР СИНХ *являются* PSPACE-*полными.*

2. *Для любого* $k \geqslant 2$ *задачи* k-БЕР СИНХ, k-БЕР СИНХ ДЛИНЫ L *и* КРАТ k-БЕР СИНХ *являются* PSPACE-*полными.*

Доказательство. Из предложения 2 следует, что задачи БЕР СИНХ, БЕР СИНХ ДЛИНЫ Lи КРАТ БЕР СИНХ лежат в классе PSPACE. Следовательно, из предложения 1 можно сделать вывод, что для любого $k \geqslant 2$ задачи k-БЕР СИНХ, k-БЕР СИНХ ДЛИНЫ L и КРАТ k-БЕР СИНХ лежат в классе PSPACE.

Из предложений 3 и 4 получаем, что задачи 2-БЕР СИНХ, 2-БЕР СИНХ ДЛИНЫ L и КРАТ 2-БЕР СИНХ являются PSPACE-трудными. Следовательно, из предложения 1 получаем, что задачи БЕР СИНХ, БЕР СИНХ ДЛИНЫ L и КРАТ БЕР СИНХ, а также задачи k-БЕР СИНХ, k-БЕР СИНХ ДЛИНЫ L и КРАТ k-БЕР СИНХ для любого $k \geqslant 2$ являются PSPACE-трудными. Следовательно, все вышеперечисленные задачи являются PSPACE-полными. Теорема доказана.

1. Černý J. Poznámka k homogénnym eksperimentom s konecnými avtomatami // Mat.-Fyz. Cas. Slovensk. Akad. Vied. 1964. Vol. 14. P. 208–216.
2. Dubuc L. Sur les automates circulaires et la conjecture de Černý // RAIRO Inform. Theor. Appl. 1998. Vol. 32. P. 21–34.
3. Eppstein D. Reset sequences for monotonic automata // SIAM J. Comput. 1990. Vol. 19. P. 500–510.
4. Kari J. Synchronizing finite automata on Eulerian digraphs // Mathematical Foundations of Computer Science 2001. [Lecture Notes in Computer Science; Vol. 2136]. B.: Springer, 2001. P. 432–438.
5. Ananichev D. S., Volkov M. V. Synchronizing monotonic automata // Developments in Language Theory. [Lecture Notes in Computer Science; Vol. 2710]. B.; Heidelberg: Springer, 2003. P. 111–121.
6. Клячко А. А., Рысцов И. К., Спивак М. А. Экстремальная комбинаторная задача, связанная с длиной синхронизирующего слова в автомате // Кибернетика. 1987. № 2. С. 16–20.
7. Ito M., Shikishima-Tsuji K. Some results on directable automata // Theory Is Forever. Essays Dedicated to Arto Salomaa on the Occasion of His 70th Birthday. [Lecture Notes in Computer Science; Vol. 3113]. B.: Springer-Verlag, 2004. P. 125–133.
8. Ito M. Algebraic theory of automata and languages. Singapore: World Scientific, 2004.
9. Martyugin P. V. Lower bounds for length of shortest carefully synchronizing words // CSR 2006, Workshop on Words and Automata, St. Petersburg, Russia.
10. Samotij W. A note on the complexity of the problem of finding shortest syncronizing words // Proc. Int. Conf. AUTOMATA. Palermo, 2007.

11. Salomaa A. Composition sequences for functions over a finite domain // Theoret. Comput. Sci. 2003. Vol. 292. P. 263–281.

12. Stockmeyer L. J., Meyer A. R. Word Problems Requiring Exponential Time // Proc. 5th Ann. ACM Symp on theory of computing, Assotiating for Computing Machinery. N. Y., 1973. P. 1–9.

13. Hopcroft J., Ulman J. Formal languages and their relation to automata. Reading, Mass.: Addison-Wesley, 1969.

Статья поступила 03.04.2008, окончательный вариант 06.05.2008