В. А. Бояршинов

ЭКВИВАЛЕНТНОСТЬ МОДЕЛЕЙ ЛОКАЛЬНЫХ ВЫЧИСЛЕНИЙ[1]

В настоящее время существует несколько моделей локальных вычислений на графах: системы переписывания графов с приоритетом [1], системы переписывания графов с запрещенными контекстами [2], локальные алгоритмы Журавлева [3, 4], (см. также [5]), сети конечных автоматов [7]. Представляет интерес вопрос сравнения классов задач, разрешимых за полиномиальное время в различных моделях локальных вычислений.

Известно [2], что модели переписывания графов с запрещенными контекстами и переписывания графов с приоритетом эквивалентны между собой. В настоящей работе доказывается, что класс задач, разрешимых за полиномиальное время с помощью систем переписывания графов с запрещеными контекстами, совпадает с классом задач, разрешимых за полиномиальное время с помощью локальных алгоритмов Журавлева и сетей конечных автоматов.

Статья является продолжением работ автора [8,9].

Предложение 1. Пусть проблема K разрешима над семейством графов $I(P)$ с помощью системы переписывания графов с запрещенными контекстами S_0 за время T_0. Тогда существует конечный автомат ω_0 такой, что проблема K разрешима над семейством графов $I(P)$ с помощью конечного автомата ω_0 за время $O(V(G) \cdot T_0)$.

Доказательство. В сети конечных автоматов моделируется повторяющийся обход графа фишкой. Каждый раз, как фишка меняет свою позицию, инициируется процесс проверки: содержит ли окрестность первого порядка фишки вхождение левой части некоторого правила переписывания из S. Если такое вхождение есть, то производится изменение состояний автоматов окрестности согласно правилу переписывания. Затем обход графа сети продолжается. Алгоритм завершает свою работу и сеть переходит в стационарное состояние, когда в течение некоторого обхода графа сети фишка не обнаружит ни одного вхождения левой части правила из S.

---

[1]Работа выполнена при финансовой поддержке Российского фонда фундаментальных исследований (грант № 01-01-794) и Министерства образования РФ.

В левой части каждого правила из S выделим вершину, которую в дальнейшем будем называть **центральной**. Центральная вершина правила выделяется согласно следующим правилам:

• Если левая часть правила является звездой с двумя и более лучами, то в качестве центральной вершины выбирается всесмежная вершина (центр) звезды.

• Если левая часть правила представляет собой ребро, то в качестве центральной вершины правила выбирается произвольный конец этого ребра.

• Если левая часть правила представляет собой одновершинный граф, то в качестве центральной вершины выбирается вершина правила.

Пусть L_V — алфавит вершинных меток, L_E — алфавит реберных меток системы переписывания с запрещенными контекстами S; $N(S)$ — число правил переписывания в S; $q(G)$ — число вершин графа G; $Q(S)$ — максимальное число вершин в левой части правила переписывания из S.

Работу системы переписывания с запрещенными контекстами S моделирует конечный автомат $R_2 = (E_2, L_2, \psi_2)$, где

$$\begin{aligned} E_2 = L_2 = \{(s,t,a,b,c,d,e) \mid\ & s \in \{0,1,\Delta\}, t \in \{A, B, CWait, \Delta\}, a \in L_V, \\ & b \in L_E, c \in \{\Delta, cktar_1 \ldots, clear_N, c_1, \ldots, c_N, \\ & \qquad draw_1, \ldots, draw_N, ver_1, \ldots, ver_N\}, \\ & d \in \{\Delta, ok, no, 1, \ldots, Q\}, e \in \{yes, no\}\}. \end{aligned}$$

Первые две позиции вектора состояния ветви автомата предназначаются для моделирования обхода в глубину графа сети; третья и четвертая позиции предназначаются для хранения меток системы переписывания S; пятая и шестая предназначены для проверки наличия вхождения левой части допустимого правила переписывания и организации непосредственно процесса переписывания допустимого вхождения; седьмая позиция предназначена для запоминания факта переписывания вхождения некоторого правила во время последнего обхода графа сети.

Функция изменения состояния автомата ψ_2 определяется множеством правил изменения состояний, которое состоит из двух групп.

Первая группа правил — правила, моделирующие обход графа сети с учетом двух замечаний: всякий раз, изменяя позицию активной вершины, передаем управление второй группе правил:

$$e^2_{i,t+1} := Wait, \forall i; e^5_{j,t+1} := c1, \forall j;$$

кроме того, производится еще один обход графа сети в том и только том случае, если в седьмой позиции состояния фишки, вернувшейся в начальную вершину, стоит отметка о моделировании применения некоторого правила переписывания в течение последнего обхода.

Вторая группа правил — правила, отвечающие за поиск и переписывание допустимого вхождения, — строится следующим образом.

Занумеруем вершины левых частей всех правил номерами из $\{1,\ldots,q(\lambda D_i)\}$. При этом центральной вершине правила приписывается номер 1; все остальные вершины нумеруются произвольным образом.

Пусть λD — помеченная звезда на n вершинах $\{v_1,\ldots,v_n\}$, где вершина v_1 смежна со всеми остальными вершинами. Сопоставим графу λD пропозициональную формулу

$$\begin{aligned} P(\lambda D) = {} & (\forall_j (e^3_{j,t} = \lambda(v_1))) \wedge (\exists i_1,\ldots,i_{n-1}((l^3_{i_1,t} = \lambda(v_2))\wedge \\ & (l^4_{i_1,t} = \lambda((v_1,v_2))) \wedge \ldots \wedge (l^3_{i_{n-1},t} = \lambda(v_n))\wedge \\ & (l^4_{i_{n-1},t} = \lambda((v_1,v_n))))), \end{aligned}$$

которую будем называть **характеристической формулой** графа λD.

Для каждого правила $r_i = (\lambda_i D_i, \lambda'_i D_i, F_{i,l},\ldots,F_{i,k_i}), i \neq N$, добавим в ψ_2 правила распознавания и переписывания вхождения λD_i, а именно:

- Если левая часть правила r_i содержит вершины $\{v_1,\ldots,v_{q(\lambda D_i)}\}$, $q(\lambda D_i) \geq 3$, то добавляются правила
  - ◇ $(e^5_{j,t} = ci, \forall j) \wedge (\neg P(\lambda D_i) \vee P(F_{i,l} \vee \ldots \vee P(F_{i,k_i})) \implies$
    $e^5_{j,t+1} := c(i+1), \forall j$
    (если нет вхождения левой части i-го правила или оно включено в запрещенный контекст, то переходим к поиску вхождения следующего правила);
  - ◇ $$\begin{aligned} (e^5_{j,t} = ci, \forall j)\wedge \ & \neg P(F_{i,l} \wedge \ldots \wedge \neg P(F_{i,k_i})\wedge \\ & (\exists j_1,\ldots j_{q(D_i)-1}((l^3_{j_1 t} = \lambda_i(v_2))\wedge \\ & (l^4_{j_1,t} = \lambda_i((v_1,v_2))) \wedge \ldots \wedge (l^3_{j_{q(D_i)-1} t} = \lambda_i(v_{q(D_i)}))\wedge \\ & (l^4_{jq(D_i)-1,t} = \lambda_i((v_1,v_{q(D-i)})))) \implies \end{aligned}$$
    $e^5_{k,t+1} := drawi, \forall k; e^6_{j1,t+1} := 2, e^6_{j_2 t+1} := 3,\ldots,\ e^6_{j_{q(D_i-1},t+1} := q(D_i)$
    (если существует вхождение левой части i-го правила и оно не включено ни в один из запрещенных контекстов, то даем соответствующим вершинам команду изменить свое состояние согласно i-му правилу переписывания);
  - ◇ $\exists j(l^5_{j,t} = drawi) \wedge (l^6_{j,t} = k) \implies$
    $e^3_{m,t+1} := \lambda'_i(v_k), \forall m; e^4_{j,t+1} := \lambda'_i((v_1,v_k))$

(если существует соседний автомат, который дает команду переписать вхождение правила r_i, при этом сообщает, что нужно сопоставить себя вершине с номером k, то изменяем свое состояние так же, как правило переписывания r_i изменяет метку k-й вершины и ребра (v_1, v_k));

$\diamond$ $\forall k(e^5_{k,t+1} = drawi \wedge \exists j_1, \dots, j_{q(D_i)-1}((e^6_{j_1,t} = 2) \wedge, \dots, \wedge (e^6_{j_{q(D_i)-1},t} = q(D_i))) \implies$

$$\begin{array}{lll} e^3_{k,t+1} := \lambda'_i(v_1), \forall k; & & e^6_{k,t+1} := \Delta, \forall k; \\ e^4_{j_1,t+1} := \lambda'_i((v_1, v_2)), & & \dots, \\ e^4_{j_{q(D_i)-1},t+1} := \lambda'_i((v_1, v_{q(D_i)})); & & e^5_{k,t+1} := \Delta, \forall k; \\ e^7_{k,t+1} := yes, \forall k; & & e^2_{k,t+1} := A, \forall k \end{array}$$

(отдав команду начать переписывание, центральная вершина правила изменяет свое состояние согласно правилу r_i; отмечает, что в течение последнего обхода графа сети было переписано по крайней мере одно вхождение и разрешает продолжить обход графа).

• Если левая часть правила r_i содержит $q(\lambda D_i) = 1$ вершину, то производятся следующие преобразования. Пусть $F_{i,l}, \dots, F_{i,m}$ — запрещенные контексты для r_i, в которых центральная вершина правила сопоставлена вершине, отличной от центра звезды (в частности, $q(F_{i,j} \geq 3, \forall i \in \{1, \dots, m\})$; $F_{i,m+1}, \dots, F_{i,k_i}$ — все остальные запрещенные контексты. Занумеруем вершины графов $F_{i,l}, \dots, F_{i,m}$ таким образом, чтобы центр звезды в каждом контексте получил номер 1, а вершина, которой сопоставлена центральная вершина правила, получила номер 2; вершины графов $F_{i,m+1}, \dots, F_{i,k_i}$ нумеруются таким образом, чтобы номер 1 получила вершина, которой сопоставлена центральная вершина правила. В определение ψ_2 добавляются следующие правила:

$\diamond$ $(e^5_{j,t} = ci, \forall j) \wedge (\neg P(\lambda_i(D_i)) \vee P(F_{i,m+1}) \vee \dots \vee P(F_{i,k_i}) \implies e^5_{j,t+1} := c(i+1), \forall j$

(если нет вхождения левой части i-го правила или оно включено в запрещенный контекст, то переходим к поиску вхождения следующего правила);

$\diamond$ $(e^5_{j,t} = ci, \forall j) \wedge P(\lambda(D_i)) \wedge \neg P(F_{i,m+1}) \wedge \dots \wedge \neg P(F_{i,k_i}) \implies e^6_{j,t+1} := 1, \forall j; e^5_{j,t+1} := ver_i, \forall j$

(если существует вхождение левой части правила r_i и оно не включено ни в один из запрещенных контекстов среди $F_{i,m+1}, \dots, F_{i,k_i}$, то переходим к проверке, не включено ли это вхождение в некоторый запрещенный контекст среди $F_{i,l}, \dots, F_{i,m}$);

◇ $\exists j \quad ((l^6_{j,t} = 1) \wedge (l^5_{j,t} = ver_i) \wedge ((P(F_{i,l}) \wedge (l^3_{j,t} = \lambda_{F_{i,1}}(v_2))$
$\wedge(l^4_{j,t} = \lambda_{F_{i,l}}((v_1, v_2)))) \vee \ldots \vee (P(F_{i,m} \wedge (l^3_{j,t} = \lambda_{F_{i,m}}(v_2))$
$\wedge(l^4_{j,t} = \lambda_{F_{i,m}}((v_1, v_2))))) \quad \Longrightarrow$
$e^6_{j,t+1} := yes$
(если сосед-автомат запросил проверить, не включен ли он в некоторый запрещенный контекст среди $F_{i,l}, \ldots, F_{i,m}$, и такое включение удалось обнаружить, то сообщить ему об этом);

◇ $\exists j \quad ((l^6_{j,t} = 1) \wedge (l^5_{j,t} = ver_i) \wedge ((\neg P(F_{i,1}) \vee \neg(l^3_{j,t} = \lambda_{F_{i,1}}(v_2))$
$\vee\neg(l^4_{j,t} = \lambda_{F_{i,l}}((v_1, v_2)))) \wedge \ldots \wedge (\neg P(F_{i,m} \vee \neg(l^3_{j,t} = \lambda_{F_{i,m}}(v_2))$
$\vee\neg(l^4_{j,t} = \lambda_{F_{i,m}}((v_1, v_2))))) \quad \Longrightarrow$
$e^6_{j,t+1} := no$
(если сосед-автомат запросил проверить, не включен ли он в некоторый запрещенный контекст среди $F_{i,1}, \ldots, F_{i,m}$, и такого включения не удалось обнаружить, то сообщить ему об этом);

◇ $\forall j(e^5_{j,t} = ver_i) \wedge \forall j(l^6_{j,t} = no) \quad \Longrightarrow$
$e^5_{j,t+1} := draw_i, \forall j; e^6_{j,t+1} := \Delta, \forall j$
(если вхождение правила r не включено ни в один запрещенный контекст среди $F_{i,l}, \ldots, F_{i,k_i}$, то перейти к переписыванию вхождения);

◇ $\exists k(l^5_{k,t} = draw_i) \quad \Longrightarrow \quad e^6_{k,t+1} := \Delta$
(если соседний автомат перешел к переписыванию вхождения, занулить вспомогательную информацию);

◇ $\forall j(e^5_{j,t} = draw_i) \quad \Longrightarrow$
$e^3_{j,t+1} := \lambda'_i(v_1), \forall j; e^5_{j,t+1} := \Delta, \forall j; e^7_{k,t+1} := yes, \forall k; e^2_{k,t+1} := A, \forall k$
(переписав вхождение правила, разрешить продолжить обход графа сети);

◇ $\forall j(e^5_{j,t} = ver_i) \wedge \exists k(l^6_{k,t} = yes) \quad \Longrightarrow$
$e^5_{j,t+1} := clear_i, \forall j; e^6_{j,t+1} := \Delta, \forall j$
(если вхождение правила r_i включено в некоторый запрещенный контекст среди $F_{i,l}, \ldots, F_{i,m}$, то дать команду всем смежным автоматам обнулить вспомогательную информацию);

◇ $\exists k(l^5_{k,t} = clear_i) \quad \Longrightarrow \quad e^6_{k,t+1} := \Delta$
(получив от соседа команду, занулить вспомогательную информацию);

◇ $\forall j(e^5_{j,t} = clear_i) \quad \Longrightarrow \quad e^5_{j,t+1} := c_{i+1}$
(после зануления вспомогательной информации перейти к поиску вхождения следующего правила).

- Если левая часть правила r_i содержит $q(\lambda D_i) = 2$ вершины (т.е.

D_i представляет собой ребро), то произвести следующие преобразования. Пусть $F_{i,1}, \ldots, F_{i,m}$ — запрещенные контексты для r_i, в которых центральная вершина правила сопоставлена вершине, отличной от центра звезды; $F_{i,m+1}, \ldots, F_{i,k_i}$ — запрещенные контексты, в которых центральная вершина правила сопоставлена центру звезды. Занумеруем вершины графов $F_{i,l}, \ldots, F_{i,m}$ таким образом, чтобы центр звезды в каждом контексте получил номер 1, а вершина, которой сопоставлена центральная вершина правила, получила номер 2; вершины графов $F_{i,m+1}, \ldots, F_{i,k_i}$ нумеруются таким образом, чтобы номер 1 получила вершина, которой сопоставлена центральная вершина правила, а номер 2 — вершина, которой сопоставлена вторая вершина из левой части правила. В определении ψ_2 добавляются следующие правила:

- $\diamond$ $(e^5_{j,t} = ci, \forall j) \wedge (\neg P(\lambda_i(D_i)) \vee P(F_{i,m+1} \vee \ldots \vee P(F_{i,k_i})) \implies e^5_{j,t+1} := c(i+1), \forall j$
  (если нет вхождения левой части i-го правила или оно включено в запрещенный контекст, то переходим к поиску вхождения следующего правила);
- $\diamond$ $(e^5_{j,t} = ci, \forall j) \wedge P(\lambda_i(D_i)) \wedge \neg P(F_{i,m+1}) \wedge \ldots \wedge \neg P(F_{i,k_i}) \implies e^6_{j,t+1} := 1, \forall j; e^5_{j,t+1} := ver_i, \forall j$
  (если существует вхождение левой части правила r_i и оно не включено ни в один из запрещенных контекстов среди $F_{i,m+1}, \ldots, F_{i,k_i}$, то переходим к проверке, не включено ли это вхождение в некоторый запрещенный контекст среди $F_{i,1}, \ldots, F_{i,m}$);
- $\diamond$ $\exists j \quad ((l^6_{j,t} = 1) \wedge (l^5_{j,t} = ver_i) \wedge ((P(F_{i,1}) \wedge (l^3_{j,t} = \lambda_{F_{i,1}}(v_2)) \wedge(l^4_{j,t} = \lambda_{F_{i,1}}((v_1, v_2)))) \vee \ldots \vee (P(F_{i,m}) \wedge (l^3_{j,t} = \lambda_{F_{i,m}}(v_2)) \wedge(l^4_{j,t} = \lambda_{F_{i,m}}((v_1, v_2))))) \implies$
  $e^6_{j,t+1} := yes$
  (если сосед-автомат запросил проверить, не включен ли он в некоторый запрещенный контекст среди $F_{i,1}, \ldots, F_{i,m}$, и такое включение удалось обнаружить, то сообщить ему об этом);
- $\diamond$ $\exists j \quad ((l^6_{j,t} = 1) \wedge (l^5_{j,t} = ver_i) \wedge ((\neg P(F_{i,1}) \vee \neg(l^3_{j,t} = \lambda_{F_{i,1}}(v_2)) \vee\neg(l^4_{j,t} = \lambda_{F_{i,1}}((v_1, v_2)))) \wedge \ldots \wedge (\neg P(F_{i,m}) \vee (l^3_{j,t} = \lambda_{F_{i,m}}(v_2)) \vee\neg(l^4_{j,t} = \lambda_{F_{i,m}}((v_1, v_2)))) \implies$
  $e^6_{j,t+1} := no$
  (если сосед-автомат запросил проверить, не включен ли он в некоторый запрещенный контекст среди $F_{i,1}, \ldots, F_{i,m}$, и такое включение не удалось обнаружить, то сообщить ему об этом);

$\diamond$ $\forall j \quad (e^5_{j,t} = ver_i) \wedge \forall k(l^3_{k,t} \neq \lambda'_i(v_2))$
$\vee(l^4_{k,t} \neq \lambda'_i((v_1, v_2))) \vee (l^6_{k,t} = yes) \quad \Longrightarrow$
$e^5_{j,t+1} := clear_i, \forall j; e^6_{j,t+1} := \Delta, \forall j$
(если любое вхождение правила r_i включено в некоторый запрещенный контекст среди $F_{i,1}, \ldots, F_{i,m}$, то дать команду всем смежным автоматам обнулить вспомогательную информацию);

$\diamond$ $\exists k(l^5_{k,t} = clear_i) \quad \Longrightarrow \quad e^6_{k,t+1} := \Delta$
(получив от соседа команду, занулить вспомогательную информацию);

$\diamond$ $\forall j(e^5_{j,t} = clear_i) \quad \Longrightarrow \quad e^5_{j,t+1} := c_{(i+1)}$
(после зануления вспомогательной информации перейти к поиску вхождения следующего правила);

$\diamond$ $\forall j(e^5_{j,t} = ver_i) \wedge \exists k(l^6_{k,t} = no) \wedge (l^3_{k,t} = \lambda_i(v_2)) \wedge (l^4_{k,t} = \lambda_i((v_1, v_2))) \Longrightarrow$
$e^5_{j,t+1} := draw_i, \forall j; e^6_{k,t+1} := 2, e^6_{j,t+1} := \Delta, \forall j \neq k$
(если некоторое вхождение правила r_i не включено ни в один запрещенный контекст среди $F_{i,1}, \ldots, F_{i,k_i}$, то перейти к переписыванию этого вхождения);

$\diamond$ $\exists k((l^5_{k,t} = draw_i) \wedge (l^6_{k,t} = \Delta)) \quad \Longrightarrow \quad e^6_{k,t+1} := \Delta$
(если соседний автомат перешел к переписыванию вхождения, не содержащего данный автомат, то занулить вспомогательную иннформацию);

$\diamond$ $\exists k((l^5_{k,t} = draw_i) \wedge (l^6_{k,t} = 2)) \quad \Longrightarrow$
$e^6_{k,t+1} := \Delta; e^3_{j,t+1} := \lambda'_i(v_2), \forall j; e^4_{k,t+1} := \lambda'_i((v_1, v_2))$
(если соседний автомат перешел к переписыванию вхождения, содержащего данный автомат, то занулить вспомогательную информацию и изменить состояние автомата согласно правилу переписывания r_i);

$\diamond$ $\forall j(e^5_{j,t} = draw_i) \wedge \exists k(e^6_{k,t} = 2) \quad \Longrightarrow$
$e^3_{j,t+1} := \lambda'_i(v_1), \forall j; \quad e^4_{k,t} = \lambda'_i((v_1, v_2)); \quad e^5_{j,t+1} := \Delta, \forall j;$
$e^7_{m,t+1} := yes, \forall m; \quad e^2_{m,t+1} := A, \forall m$
(переписав вхождение правила, разрешить продолжить обход графа сети).

• Построение правил изменения состояния автомата по правилу переписывания r_N производится аналогично тому, как описано выше, за единственным исключением: вместо перехода к поиску вхождений следующего правила возобновляется обход графа сети.

Сеть конечных автоматов R, граф которой $G(R)$ лежит в $I(P)$, и каждой вершине которой сопоставлена копия автомата $\Omega_0 = (E_2, L_2, \psi_2)$, всегда переходит в стационарное состояние, поскольку моделирует допустимую цепь переписывания системы переписывания с запрещенными контекстами S_0 на графе из $I(P)$.

Поскольку на один обход графа сети требуется $O(V(G))$ тактов работы сети, а во время каждого обхода моделируется применение как минимум одного правила переписывания, то сеть автоматов перейдет в стационарное состояние не более чем через $O(V(G) \cdot N_0)$ тактов работы.

Предложение 2. Пусть проблема K разрешима над семейством графов $I(P)$ с помощью локального параллельного алгоритма Журавлева A_0 за время T_0. Тогда существует конечный автомат Ω_0 такой, что проблема K разрешима над семейством графов $I(P)$ с помощью конечного автомата Ω_0 за время $O(T_0)$.

Доказательство. Пусть $P_1^v, P_2^v, \ldots, P_m^v$ — предикаты, определенные на вершинах графа алгоритмом $A_0; P_1^e, P_2^e, \ldots, P_n^e$ — предикаты, определенные на ребрах графа алгоритмом A_0.

Построим по алгоритму A_0 эквивалентный ему конечный автомат $\Omega_0 = (E_0, \psi_0)$ следующим образом.

Заметим, что между множеством инциденторов графа и множеством ветвей конечного автомата существует естественное взаимооднозначное соответствие; обозначим его через ϕ.

В качестве множества состояний ветвей автомата возьмем множество $E_0 = \{(a_1, \ldots, a_m, b_1, \ldots, b_n) \mid a_i, b_j \in \{1, 0, \Delta\}\}$. Другими словами, состояние ветви автомата есть вектор, являющийся конкатенацией информационных векторов для соответствующей вершины и инцидентного ей инцидентора графа.

Функция изменения состояний автомата Ω_0 строится по функциям изменения значений предикатов алгоритма A_0 следующим образом:

- $e_{(v,b),t+1} = (\varphi_1^v(\phi(S_1((v,b))) \mid_{\phi(v)}), \varphi_2^v(\phi(S_1((v,b))) \mid_{\phi(v)}), \ldots,$
  $\varphi_m^v(\phi(S_1((v,b))) \mid_{\phi(v)}), \varphi_1^e(\phi(S_1((v,b))) \mid_{\phi(v,b)}),$
  $\varphi_2^e(\phi(S_1((v,b))) \mid_{\phi(v,b)}), \ldots, \varphi_n^e(\phi(S_1((v,b))) \mid_{\phi(v,b)}))$.

Пусть (v, b) — ветвь конечного автомата, $e = phi((v, b))$ — соответствующий ей инцидентор графа. В качестве начального состояния ветви (v, b) возьмем следующее:

$$(P_{0,1}^v(v), P_{0,2}^v(v), \ldots, P_{0,m}^v(v), P_{0,1}^e(e), P_{0,2}^e(e), \ldots, P_{0,n}^e(e)).$$

Другими словами, начальное состояние ветви автомата есть конкатенация начальных информационных векторов для соответствующей вершины и инцидентного ей инцидентора графа.

Построенный автомат $\Omega_0 = (E_0, \psi_0)$ является искомым, поскольку его действие на графе идентично действию локального параллельного алгоритма Журавлева A_0.

Предложение 3. Пусть проблема K разрешима над семейством графов $I(P)$ в модели сетей конечных автоматов за время T_0. Тогда существует локальный параллельный алгоритм Журавлева, дающий решение проблемы K над семейством графов $I(P)$ за время $O(T_0)$.

Доказательство. Пусть конечный автомат $\Omega_0 = (E_0, \phi_0)$ дает решение проблемы K над семейством графов $I(P)$ за время $O(T_0)$. Построим по нему локальный параллельный алгоритм Журавлева A_0, также дающий решение этой проблемы над семейством графов $I(P)$ за время $O(T_0)$.

Заметим, что между множеством инциденторов графа и множеством ветвей конечного автомата существует естественное взаимооднозначное соответствие; обозначим его через ϕ.

Определим на инциденторах графа следующие предикаты:

$$P_1^0, P_2^0, \ldots, P_{|E_0|}^0.$$

При этом значение i-го предиката на инциденторе графа e равно истине в том и только том случае, если соответствующая ему ветвь $\phi(e)$ имеет метку $a_i \in E_0$. В любой момент времени работы локального алгоритма Журавлева A_0 и на любом инциденторе графа будет выполняться следующее условие: ровно один предикат, определенный на данном инциденторе, имеет значение равное **истина**. Значение всех остальных предикатов на данном инциденторе равно **ложь**. Таким образом, по информационному вектору $I(e)$ инцидентора e однозначно восстанавливается метка ветви $\phi(e)$.

Пусть v — некоторая вершина графа, e — инцидентор этой вершины. Определим функции $\varphi_1^0, \ldots, \varphi_{|E_0|}^0$ изменения значений предикатов следующим образом:

- Если $\psi_0(\phi(S(v))) \mid_{\phi(e)} = Id$, то

$$\begin{aligned}
P_{t+1,1}^0(e) &= \varphi_{t+1,1}^0(e) &&:= P_{t,1}^0(e),\\
P_{t+1,2}^0(e) &= \varphi_{t+1,2}^0(e) &&:= P_{t,2}^0(e), \ldots,\\
P_{t+1,|E_0|}^0(e) &= \varphi_{t+1,|E_0|}^0(e) &&:= P_{t,|E_0|}^0(e).
\end{aligned}$$

- Если $\psi_0(\phi(S(v))) \mid_{\phi(e)} \neq Id, \psi_0 \mid_{\phi(e)}$ предписывает изменить состояние ветви $\phi(e)$ на $a_i \in E_0$ и $P_{t,k}^0(e) = true$, то произвести следующие изменения значений предикатов:

$$
\begin{array}{ll}
P^0_{t+1,k}(e) := false, & P^0_{t+1,i}(e) := true; \\
P^0_{t+1,j}(e) := P^0_{t,j}(e), & \forall j \neq i, k.
\end{array}
$$

Пусть v — некоторая вершина графа, e — инцидентор этой вершины, $\phi(e)$ — ветвь конечного автомата, соответствующая инцидентору e, которая имеет начальную метку $a_i \in E_0$. Тогда начальные значения предикатов на инциденторе e устанавливаются следующим образом: $P^0_{0,i}(e) := true, P^0_{0,j}(e) := false, \forall j \neq i$.

Построенный алгоритм Журавлева является искомым автоматом, решающим проблему K, так как его поведение на графе неотличимо от поведения сети конечных автоматов Ω_0.

Предложение 4. Пусть проблема K разрешима над семейством графов $I(P)$ с помощью конечного автомата Ω_0 за время T_0. Тогда существует система переписывания графов с запрещенными контекстами S_0 такая, что проблема K разрешима над семейством графов $I(P)$ с помощью S_0 за время $O(|V(G)| \cdot T_0)$.

Доказательство. Идея доказательства заключается в следующем. С помощью системы переписывания графов с запрещенными контекстами производится повторяющийся обход графа в ширину. Во время каждого такого обхода моделируется один такт работы сети конечных автоматов Ω_0. Алгоритм закончит свою работу, когда произойдет первый обход графа, в течение которого метка ни одного инцидентора графа не была изменена.

Каждый раз, когда вершина графа затронута новым обходом в ширину, в ней инициализируется процесс моделирования одного такта работы конечного автомата Ω_0 в данной вершине. Он заключается в изменении меток всех инциденторов данной вершины согласно предписаниям функции изменения состояний ψ_0 для Ω_0.

Метка каждого инцидентора будет состоять из двух частей. Первая часть метки предназначается для результатов вычислений, проделанных во время последнего обхода графа в ширину; вторая предназначается для хранения результатов вычислений, проделанных на предыдущем обходе. При окончании очередного обхода информация из первой части метки переписывается во вторую, после чего первая часть зануляется.

Осталось только показать способ моделирования одного такта работы автомата Ω_0 в данной вершине.

Рассмотрим общий вид правила из функции изменения состояний автомата ψ_0 (в квадратных скобках указана необязательная компонента).

Если

$$(\exists i_1, i_2, \ldots, i_k, i_{k+1}, \ldots, i_s|\ e_{(i_1,t)} = a_{i_1}, \ldots, e_{(i_k,t)} = a_{i_{k+1},t} = a_{i_{k+1}}, \ldots,$$
$$l_{i_s,t} = a_{i_s})\wedge$$
$$\Big[\nexists j_1, j_2, \ldots, j_r \mid (e_{(j_1,t)} = b^1_{j_1}, \ldots, e_{(j_k,t)} = b^1_{j_k},$$
$$l_{(j_{k+1},t)} = b^1_{j_{k+1}}, \ldots, l_{(j_r,t)} = b^1_{j_r}) \bigvee \ldots$$
$$\bigvee(e_{(j_1,t)} = b^d_{j_1}, \ldots, e_{(j_k,t)} = b^d_{j_k},$$
$$l_{(j_{k+1},t)} = b^d_{J_{k+1}}, \ldots, l_{(j_r,t)} = b^d_{j_r})\Big],$$

то $e_{i_1,t+1} := c_{i_1}, \ldots e_{i_s,t+1} := c_{i_s} (a_l, b_l, c_l \in E_0)$.

Для каждого такого правила из ψ_0 добавляется одно правило переписывания с d запрещенными контекстами (указаны только вторые части меток).

Так как число тактов работы сети конечных автоматов равно T_0, а на моделирование одного такта требуется не более чем $O(V_G)$, то длина максимально возможной цепи переписывания для системы переписывания графов S_0 не превышает $O(V_G \cdot T_0)$.

СПИСОК ЛИТЕРАТУРЫ

1. **Billaud M., Lafon P., Metivier Y., Sopena E.** Graph rewriting systems with priority: definitions and applications. — Univ. Bordeaux 1: Intern. Rep. N 8909, 1989.
2. **Litowsky I., Metivier Y., Sopena E.** Different local controls for graph relabelling systems // Math. Syst. Theory. — 1995. — Vol. 28. — P. 41–65.
3. **Журавлев Ю.И.** Локальные алгоритмы вычисления информации. 1 // Кибернетика. — 1965. — N 1. — С. 12–19.
4. **Журавлев Ю.И.** Оценки сложности локальных алгоритмов для некоторых экстремальных задач на конечных множествах // ДАН СССР. — 1964. — Т. 158, N 5. — С. 1018–1021.
5. **Евстигнеев В.А.** О некоторых свойствах локальных алгоритмов на графах // Комбинаторно-алгебраические методы в прикладной математике. — Горький, 1983. — С. 72–105.
6. **Евстигнеев В.А.** Локальные алгоритмы и проблема установления конца вычислений в распределенных системах // Комбинаторно-алгебраические и вероятностные методы дискретного анализа. — Горький, 1989. — С. 55–63.
7. **Rosenstiehl P., Fiksel J.R., Holliger A.** Intelligent graphs: networks of finite automata capable of solving graph problems // J. Graph Theory and Computing, 1972.
8. **Евстигнеев В.А., Бояршинов В.А.** Распознавание хордальных графов в рамках распределения модели вычислений // Проблемы систем информатики и программирования. — Новосибирск, 1999. — С. 107—129.
9. **Бояршинов В.А.** Распознавание интервального порядка на базе локальных вычислений // Там же. — С. 130–145.