УДК 004.942 DOI: 10.17587/it.24.274-288

А. Н. Родионов, д-р техн. наук, вед. науч. сотр., ran@newmail.ru,
Вычислительный центр ДВО РАН, г. Хабаровск

Метод и алгоритм структурного анализа достижимости узлов в сетях потоков работ

Постоянно растущая сложность деловых процессов, составляющих базис современной организации и управления сложными динамическими системами, объективно влечет за собой усложнение соответствующих моделей, известных как модели потоков работ (WorkFlow-сети). Разработано и используется множество методов обнаружения разнообразных дефектов, присущих таким моделям. В работе излагается метод, позволяющий определить достижимость отдельных узлов сети, не прибегая к стандартной процедуре преобразования WF-сети в сеть Петри. В методе используются элементы алгебры логики и ряд характеристик сети, отражающих ее организацию для установления полной, частичной достижимости или недостижимости узлов, появляющихся в экземплярах (реализациях) конкретной сети потоков работ.

Ключевые слова: *сеть потоков работ, верификация, анализ достижимости, бизнес-процесс*

Введение

Информационные системы (ИС) поддержки потоков работ постепенно вытесняют ИС, ориентированные на регистрацию фактов. Функциональная часть таких систем представлена совокупностью структурных моделей (моделями потоков работ, WorkFlow-сетями), изображающих процесс в виде альтернативных последовательностей действий.

На сегодняшний день создано и используется множество языков моделирования, некоторые из которых, такие, например, как eEPC и IDEF3, стали стандартом де-факто. Все они представляют процессы в виде графов, которые в обязательном порядке включают логические узлы, моделирующие альтернативные реализации процессов, и узлы-действия, которые соотносятся с определенными управленческими директивами.

Аномалии (дефекты) WorkFlow (WF)-сетей достаточно хорошо изучены и исследованы. WF-сеть считается бездефектной, если выполняется ряд условий: ее завершение гарантировано, в ней отсутствуют "висячие" узлы, нет тупиков и активных блокировок [1, 2]. Разработано множество методов, позволяющих обнаружить все перечисленные дефекты. Среди них имеет смысл выделить две группы. Первую группу образуют методы, использующие сети Петри в качестве аналогов исходных сетей [1—9]. Теория сетей Петри включает алгоритмы обнаружения (но не исправления) потенциальных дефектов. Другая группа методов использует логику высказываний [10—12], что более естественно, если принять в расчет, что логические операции WF-сетей — это аналоги булевых функций конъюнкции, дизъюнкции и сложения по модулю 2.

Достижимость процесса трактуется как способность (свойство) всех его экземпляров завершиться в конечной точке. Бездефектная WF-сеть — гарантированно достижима. Тем не менее WF-сеть может оказаться достижимой и при наличии некоторых дефектов.

На практике достаточно распространены ситуации, когда один и тот же процесс может корректно завершиться не в одной, а в разных точках (множество терминальных вершин включает более одного элемента). Кроме того, для части реализаций процесса важно, чтобы были достигнуты не только конечные, но и промежуточные узлы, где, как и в терминальных, могут формироваться определенные результаты [13], на получение которых ориентирован процесс. (В ИС эти результаты представляются в виде фактов.) В этой связи задача достижимости приобретает иную, расширенную трактовку. Процесс будет считаться достижимым, если в результате его реализации могут быть достигнуты некоторые наперед заданные промежуточные вершины.

Постановка и решение такой задачи вызывают определенные затруднения, если предварительно не получены данные, касающиеся достижимости отдельных узлов.

Настоящая работа нацелена на разработку метода, позволяющего получить оценку достижимости всех узлов-действий для произвольного потока работ, который задан в виде WF-сетей.

Ниже перечислены обозначения, наиболее часто встречающиеся в работе:

$\bar{c}_i$ — разворачивающий узел графа;

$\underline{c}_j$ — сворачивающий узел графа;

a_k — функциональный узел графа;

v_n — узел графа;

$\bar{\bar{c}}_r$ — родительский узел v_n-го узла сети;

S_{v_n} — статус v_n-го узла графа;

$S_{(\ldots,\ \ldots)}$ — статус дуги графа;

$DomF_{\underline{c}}$ — область определения булевой функции сворачивающего соединения;

$\Delta_{\underline{c}}$ — параметр свертки $\underline{c}$ -соединения;

$f_{\bar{C}L}(\bar{c}_i)$ — логическая операция разворачивающего узла;

$f_{\bar{C}L}(\underline{c}_j)$ — логическая операция сворачивающего узла.

1. Модель бездефектной сети потоков работ. Постановка задачи достижимости

Сеть потоков работ — это ациклический связный ориентированный граф WF = (V, E), где V — множество вершин (узлов), а E — множество ребер (дуг). В WF-графе различают вершины трех типов: действий (функциональных узлов, a-узлов), образующих множество $A = \{a_k\}_1^a$; разворачивающих узлов ($\bar{c}$ -узлов), составляющих множество $\bar{C} = \{\bar{c}_i\}_1^{\bar{c}}$; сворачивающих узлов ($\underline{c}$ -узлов), входящих во множество $\underline{C} = \{\underline{c}_j\}_1^{\underline{c}}$. По определению, $V = A \cup \bar{C} \cup \underline{C}$. Каждому сворачивающему и разворачивающему узлу ставится в соответствие определенная логическая операция (класс соединения): AND ($\wedge$) — логическое И; OR ($\vee$) — исключающее ИЛИ; XOR ($\veebar$) — включающее ИЛИ. Формально это означает, что имеют место функциональные зависимости: $f_{\underline{C}L}: \underline{C} \to L$ и $f_{\bar{C}L}: \bar{C} \to L$, где $L = \{\wedge, \vee, \veebar\}$, исключающие появление в WF-графе логических узлов, которым не назначены логические операции.

Обозначим через $\cdot v$ — множество дуг, входящих в вершину v. Соответственно, $v\cdot$ — множество дуг, исходящих из вершины v. Среди множества вершин выделим две вершины: начальную — a_s, для которой $\cdot a_s = \varnothing$ и $a_s\cdot \neq \varnothing$, и терминальную — a_f, для которой $\cdot a_f = \varnothing$ и $a_f\cdot \neq \varnothing$. Присутствие a_s и a_f в WF-сети обязательно. Выполнение данного условия гарантирует наличие единственной вершины, обозначающей "вход", и единственной вершины, ассоциирующейся с "выходом". Отсюда, любой граф потоков работ будет всегда включать как минимум две вершины — начальную и терминальную.

Разворачивающие и сворачивающие узлы моделируют альтернативы реализаций действий. Исходя из этого, логично ограничиться определенными классами отношений, которые могут устанавливаться между узлами сети, относящихся к подмножествам A, $\bar{C}$ и $\underline{C}$. Наглядное представление о таких отношениях дают схемы, показанные на рис. 1.

Узлы-действия могут быть связаны либо с действиями, либо с разворачивающими или сворачивающими узлами. То же самое справедливо и в отношении разворачивающих и сворачивающих узлов. Штриховые линии показывают, что в конкретном экземпляре отношения могут принимать участие только узлы, принадлежащие одному из типов: A, $\bar{C}$ или $\underline{C}$.

В данной модели (вследствие включения в нее исходного и терминального узлов и обязательности непустых значений $|\cdot v|$ и $|v\cdot|$ для $\forall v \in V$) исключено появление изолированных и дополнительных терминальных вершин. Тем не менее ни одна из моделей потоков работ не гарантирует отсутствия тупиков, когда процесс может завершиться в любой точке, не достигнув a_f. В то же время, другой элемент некорректности в виде "активной блокировки", представляющей собой бесконечный цикл, в настоящей модели отсутствует, так как по определению WF — это ациклический граф.

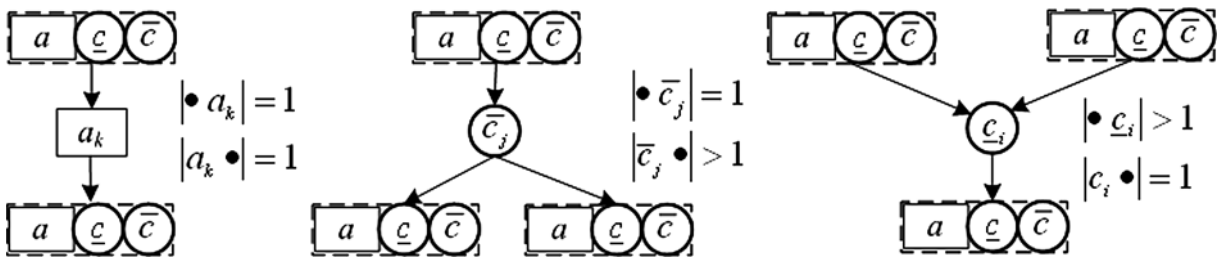


Рис. 1. Отношения между узлами сетей потоков работ

Каждой WF-сети после исключения из нее логических узлов соответствуют несколько "стандартных" сетей, в которых присутствуют только узлы-действия. Такие графы будем далее именовать графами реализации или экземплярами сети.

Для любого узла сети можно указать развернувший его узел (родительский узел — РУ). Сделать это можно следующим образом. Если к графу добавить фиктивный разворачивающий узел $\bar{c}_d$, предшествующий a_s, и поменять направления всех дуг в графе на противоположные, то каждому узлу сети, кроме $\bar{c}_d$, может быть поставлен в соответствие такой узел. Для части узлов-действий и разворачивающих узлов таким узлом окажется первый разворачивающий узел, лежащий на пути, который ведет от этих узлов к $\bar{c}_d$.

Для сворачиваемых узлов $\underline{c}_j$, являющихся началом более чем одного пути, их родительскими узлами будут узлы, являющиеся конечными на всех путях, берущих начало в $\underline{c}_j$.

Таким образом, в представленной модели, кроме множества функциональных зависимостей, отражающих отношения между узлами: $f_{a\bar{c}} : \underline{A} \to \bar{C}$; $f_{a\underline{c}} : A \to \underline{C}$; $f_{aa} : A \to A$; $f_{\bar{c}\,\bar{c}} : \bar{C} \to \bar{C}$; $f_{\bar{c}\underline{c}} : \bar{C} \to \underline{C}$; $f_{\underline{c}\underline{c}} : \underline{C} \to \underline{C}$ (последние две функции являются многозначными), присутствуют функциональные зависимости, показывающие "принадлежность" произвольного узла непосредственно или опосредованно развернувшему его родительскому узлу: $f_{a\bar{\bar{c}}} : A \to \bar{C}$; $f_{\bar{c}\bar{\bar{c}}} : \bar{C} \to \bar{C}$; $f_{\underline{c}\bar{\bar{c}}} : \underline{C} \to \bar{C}$. То же самое можно записать по-другому: $f_{a\bar{\bar{c}}}(a_k) = \bar{\bar{c}}_r$; $f_{\bar{c}\bar{\bar{c}}}(\bar{c}_i) = \bar{\bar{c}}_r$; $f_{\underline{c}\bar{\bar{c}}}(\underline{c}_j) = \bar{\bar{c}}_r$, где $\bar{\bar{c}}_r$ — родительский узел соответственно узлов a_k, $\bar{c}_i$ и $\underline{c}_j$. Очевидно, что и $\bar{\bar{c}}_r$, и $\bar{c}_i \in \bar{C}$.

Наглядное представление о родительских узлах графа потоков работ дает фрагмент WF-сети, показанный на рис. 2. Пунктирными линиями обозначены ссылки на родительские узлы.

Разрабатываемый метод — дальнейшее развитие группы методов, использующих логику высказываний для обнаружения дефектов в сетях потоков работ. Он предназначен для определения полной или частичной достижимости или недостижимости функциональных узлов WF-сетей посредством последовательного посещения каждого из узлов. Вводится характеристика S_{v_n}, именуемая статусом узла v, которая может принимать одно из трех значений: ‘+’, ‘±’ и ‘−’. Если S_{v_n} = ‘+’, то это означает, что узел v появится во всех экземплярах сети. Если статус узла получает значение ‘±’, то узел достижим только в части экземпляров сети. Значение ‘−’ свидетельствует о полной недостижимости узла. Несмотря на то что разворачивающие и сворачивающие узлы никогда не появляются в реализациях сети, для них также вычисляется статус, так как эти узлы наравне с функциональными узлами могут быть задействованы в сворачивающих и разворачивающих операциях.

2. Статусы узлов WF-сети и алгоритм их вычисления

Начальный узел сети a_s всегда имеет статус ‘+’, S_{a_s} = ‘+’. Если за a_s следуют узлы a или $\underline{c}$, то их статусы также будут иметь значение ‘+’ (наследовать статус предшествующего узла). Статус узла может измениться только в двух случаях — когда узлу предшествуют либо разворачивающий, либо сворачивающий узел, с которым соотнесена определенная логическая операция. Это можно увидеть на рис. 3, когда логические операции разворачивающего и сворачивающего узлов меняются с $\wedge$ на $\vee$.

Чтобы получить набор формальных правил, позволяющих вычислить статусы узлов в процессе последовательного посещения вершин графа, будем также оценивать возможность появления или непоявления дуг посредством нахождения статусов последних. Набор соответствующих правил

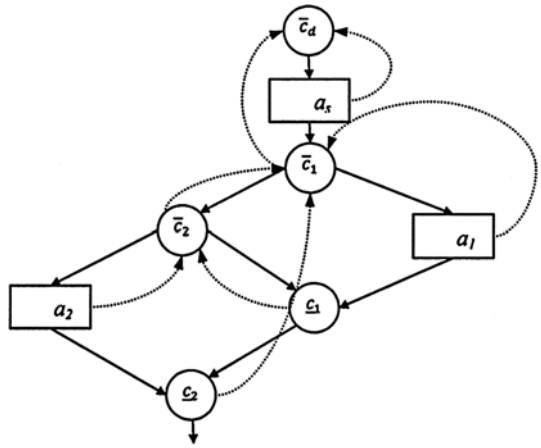


Рис. 2. Родительские узлы WF-сети

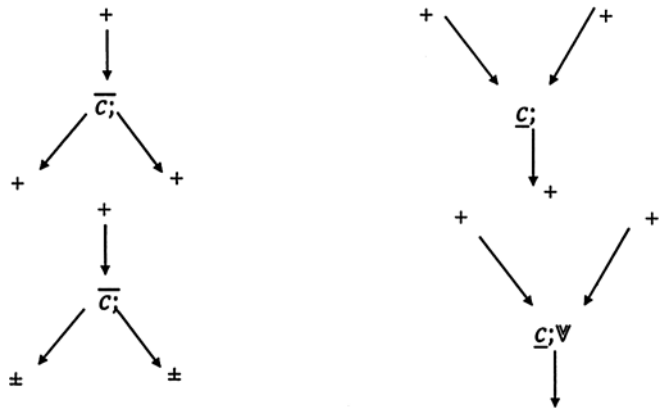


Рис. 3. Изменение статусов узлов WF-сети

для функциональных и сворачивающих узлов, а также дуг графа будет иметь следующий вид:

$$S_{\langle a_k, a_{k+1}\rangle} = S_{a_k}, \quad (1)$$

где $S_{\langle a_k, a_{k+1}\rangle}$ — статус дуги $<a_k, a_{k+1}>$, соединяющей вершину a_k с a_{k+1};

$$S_{\langle a_k, \bar{c}_i\rangle} = S_{a_k}; \quad (2)$$

$$S_{\langle a_k, \underline{c}_j\rangle} = S_{a_k}; \quad (3)$$

$$S_{\langle \bar{c}_i, a_k\rangle \vee \langle \bar{c}_i, \bar{c}_{i+1}\rangle \vee \langle \bar{c}_i, \underline{c}_j\rangle} = \begin{cases} '+', S_{\bar{c}_i} = '+' \wedge f_{\bar{C}L}(\bar{c}_i) = '\wedge' \\ '\pm', S_{\bar{c}_i} = '\pm' \vee (S_{\bar{c}_i} = '+' \wedge (f_{\bar{C}L}(\bar{c}_i) = \\ = '\Vee' \vee f_{\bar{C}L}(\bar{c}_i) = '\vee')); \\ '-', S_{\bar{c}_i} = '-' \end{cases} \quad (4)$$

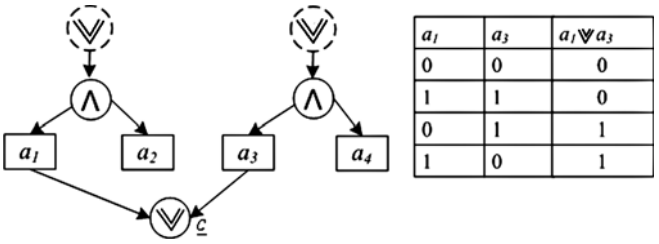
$$S_{a_k} = S_{\langle a_{k-1}, a_k\rangle} \vee S_{\langle \bar{c}_i, a_k\rangle} \vee S_{\langle \underline{c}_j, a_k\rangle}; \quad (5)$$

$$S_{\bar{c}_i} = S_{\langle \underline{c}_j, \bar{c}_i\rangle} \vee S_{\langle a_k, \bar{c}_i\rangle} \vee S_{\langle \bar{c}_{j-1}, \bar{c}_i\rangle}. \quad (6)$$

Статусы дуг, исходящих из узлов-действий, будут совпадать со статусом исходной вершины-действия (1—3). Значения статусов дуг, берущих начало в $\bar{c}$ -узлах, определятся на основании статусов разворачивающих узлов и логических операций этих узлов (4). Статусы a и $\bar{c}$ -узлов могут быть вычислены на основании статусов входящих в них дуг (5, 6).

Вычисление статусов сворачивающих соединений представляет собой более сложную задачу. Обратимся к методу [11], позволяющему провести проверку на табуированность $\underline{c}$ -узлов в WF-сетях.

Данный метод на основании значения логической операции, которое присваивается сворачивающему узлу, и всех возможных сочетаний появления или непоявления действий (или других категорий узлов), участвующих в свертке, дает возможность определить — достижимы или нет $\underline{c}$ -узлы во всех или отдельных реализациях сети. С этой целью используются булевы функции конъюнкции, дизъюнкции и сложения по модулю два, которые соответствуют логическим операциям $\wedge$, $\vee$ и $\Vee$. Число переменных таких функций равно числу сворачиваемых узлов, а область определения $DomF_{\underline{C}} = V_1 \times V_2 \times \ldots V_n \times \ldots \times V_m$, где $F_{\underline{C}}$ — вид булевой функции сворачивающего узла; m — число сворачиваемых узлов. Переменная принимает значение 1, если узел появляется в конкретной реализации сети, и значение 0 — в противном случае. В WF-сетях $V_n = \{1, 0\}$ или $V_n = \{1\}$. На представленном ниже фрагменте сети (рис. 4) демонстрируется суть метода.



a_1	a_3	$a_1 \Vee a_3$
0	0	0
1	1	0
0	1	1
1	0	1

Рис. 4. Анализ на табуированность сворачивающего соединения

Состав $DomF_{\underline{C}}$ зависит от двух факторов: класса соединения, которым развернуты действия (в представленном варианте сети и a_1, и a_3 предшествует логическое $\wedge$ — $\wedge$-класс), а также типа входа (вход $\wedge$ или вход $\Vee$) разворачивающего соединения. Вход $\wedge$ показывает, что на всех путях, ведущих к разворачивающему узлу, могут присутствовать только разворачивающие узлы со значением $\wedge$ либо вообще отсутствовать какие-либо разворачивающие соединения. Это также означает, что разворачивающий узел со значением входа $\wedge$ появится во всех экземплярах сети. Вход $\Vee$ свидетельствует об обратном — на одном или нескольких путях должен присутствовать разворачивающий узел с логическими значениями $\Vee$ или $\vee$.

Посредством введенной характеристики S_v метод проверки на табуированность $\underline{c}$ -узлов может быть упрощен и встроен в процедуру вычисления статусов всех категорий узлов, присутствующих в WF-сетях. Для нахождения $S_{\underline{c}}$ потребуется последовательно выполнить следующие действия.

Сначала на основании значений статусов, которые имеют сворачиваемые узлы, сформировать $DomF_{\underline{C}}$. Очевидно, что если S_{v_n} = '+', где S_{v_n} — статус сворачиваемого узла, то $v_n = 1$. В случае, когда S_{v_n} = '±', переменная v_n принимает значение или 0, или 1. При S_{v_n} = '–' всегда $v_n = 0$.

Следующим шагом должно стать вычисление всех значений булевой функции сворачивающего соединения $\underline{c}$.

И, наконец, для вычисления $S_{\underline{c}}$ достаточно воспользоваться простой алгоритмической процедурой, использующей вспомогательный параметр I:

$$I = 0; \textit{ for } k \textit{ from } 1 \textit{ to } |DomF_{\underline{C}}| \; I = \\ = I + F_{\underline{C}}(x_1, \ldots, x_n),$$

где $F_{\underline{C}}(x_1, \ldots, x_n)$ — булева функция соединения $\underline{c}$, сворачивающего m узлов WF-сети.

Получив подобным образом I, можно найти $S_{\underline{c}}$:

$$S_{\underline{c}} = \begin{cases} '-', 0 < I < |DomF_{\underline{c}}| \\ '\pm', I = |DomF_{\underline{c}}| \\ '-', I = 0. \end{cases}$$

У только что изложенного метода есть один недостаток. $DomF_{\underline{C}}$, найденная только на основании статусов сворачивающих узлов, может оказаться некорректной, так как не учитывает ряд важных структурных особенностей организации WF-сетей. Отразим эти особенности посредством выделения и классификации специфических подграфов, которые будем называть классами соединений.

Под классами соединений далее будем подразумевать сворачиваемые подграфы WF-сети, в которых узел $\underline{c}$ выступает в качестве терминального узла. В такие подграфы включаются узлы, оказывающие влияние на состав области определения функции соединения $\underline{c}$. Кроме непосредственно самого узла $\underline{c}$ и сворачиваемых узлов в такие подграфы входят родительские вершины, посредством которых были опосредованно развернуты сворачиваемые узлы, и узлы, непосредственно развернутые родительскими узлами. (Опосредованно означает, что между родительскими и сворачиваемыми узлами могут присутствовать промежуточные узлы.)

Обобщим и уточним сказанное, выделив, классифицировав и проанализировав все допустимые классы сворачиваемых конструкций WF-сетей на предмет получения корректных $DomF_{\underline{C}}$.

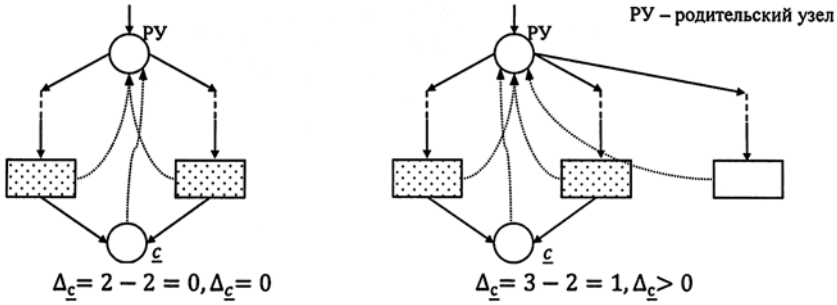


Рис. 5. Классы подграфов локальной свертки

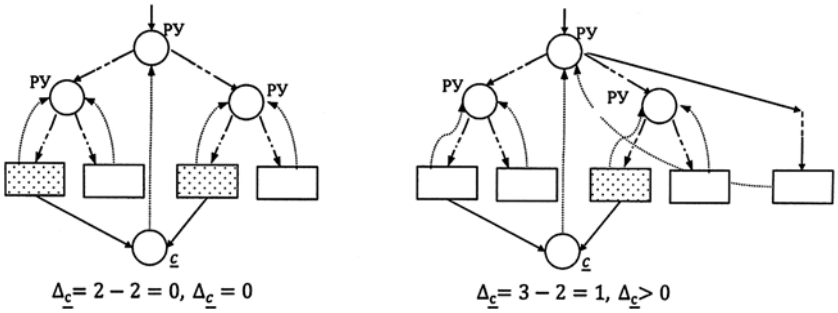


Рис. 6. Классы подграфов комбинированной свертки без элементов вложенности (простой ПКС)

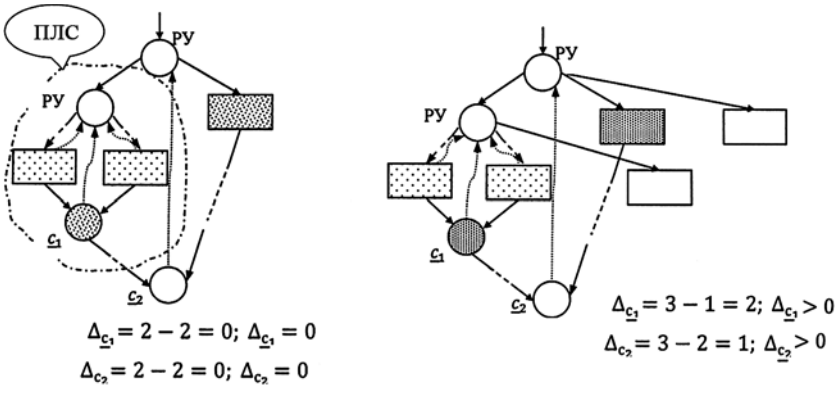


Рис. 7. ПКС с вложенными ПЛС

3. Классы и экземпляры сворачиваемых подграфов

Чтобы определиться с классами сворачиваемых подграфов, проведем их классификацию, выделив два признака, которые позволяют учесть все допустимые конфигурации подобных конструкций. Первый признак — это принадлежность всех сворачиваемых узлов одному или нескольким соединениям, которые их развернули. Второй признак призван показать, все ли узлы, развернутые одним соединением, свернуты другим. Введем параметр $\Delta_{\underline{c}}$, который количественно представит значение этого признака. Для его вычисления воспользуемся следующим выражением: $\Delta_{\underline{c}} = N_W - N_U$, где N_W — число узлов, которые были развернуты узлом, являющимся родительским для узла $\underline{c}$; N_U — число узлов, сворачиваемых узлом $\underline{c}$. Здесь важно иметь в виду следующее. Если $W_{\bar{c}_i}$ — множество узлов, которые развернуты соединением $\bar{c}_i$, то $N_w = |W_{\bar{c}_i}|$, а $N_U = |U|$, $U \subseteq W$. Другими словами, N_w и N_U — это всегда характеристики, в данном случае мощности, двух множеств W и U, одно из которых является подмножеством другого.

Комбинации значений перечисленных признаков позволяют выделить классы сворачиваемых подграфов.

Подграфы, в которых все сворачиваемые узлы и непосредственно сам сворачивающий узел $\underline{c}$ ссылаются на один и тот же родительский узел (рис. 5), назовем подграфами локальной свертки (ПЛС), а подграфы, содержащие больше одного РУ, — подграфами комбинированной свертки (ПКС). Среди ПКС будем различать ПКС, не содержащие вложенных сворачивающих подграфов (рис. 6), и ПКС, включающие таковые. В свою очередь, вложенными могут оказаться как

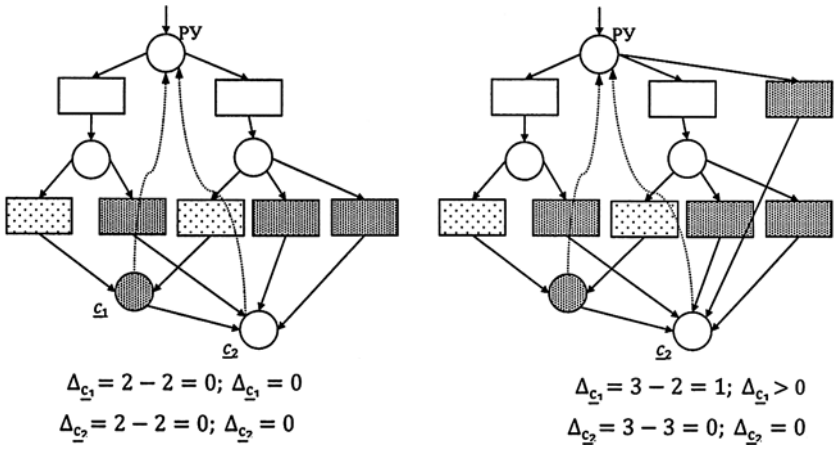


Рис. 8. ПКС с вложенными ПКС

ПЛС (рис. 7), так и ПКС (рис. 8). При этом число уровней вложенности неограниченно.

Подграфы с узлами $\underline{c}_1$ на рис. 7 считаются "вложенными" в подграфы с узлами $\underline{c}_2$ на том основании, что существует путь из $\underline{c}_1$ в $\underline{c}_2$.

Согласно второму классификационному признаку выделяются подграфы с $\Delta_{\underline{c}} = 0$ и подграфы, в которых $\Delta_{\underline{c}} > 0$.

На представленных схемах каждый узел ссылается на свой родительский узел, для чего задействованы пунктирные линии.

Проведем анализ полученных подграфов на предмет нахождения области определения функции сворачивающего узла $\underline{c}$ для всех возможных экземпляров таких подграфов. Экземпляры подграфа будут различаться статусами разворачиваемых узлов (родительских узлов) и логическими функциями этих узлов. Полученные результаты представим графически в виде таблиц истинности, исключив из последних строки, которые не могут в них появиться вследствие влияния статусов сворачиваемых и разворачивающих узлов, а также вида логической функций РУ. Число возможных сочетаний статусов и логических функций (три значения статуса и три разновидности логических функций) будет равно девяти экземплярам, что составит область анализа.

4. Экземпляры подграфов локальной свертки

Для экземпляров ПЛС будет характерна следующая картина (рис. 9). Каждый из изображенных экземпляров содержит дополнительный узел a_k, отличный от $\underline{c}$. Для каждого экземпляра приведены по две таблицы истинности: одна для конфигурации с $\Delta_{\underline{c}} = 0$, другая с $\Delta_{\underline{c}} > 0$.

Дадим несколько важных пояснений к содержанию представленных таблиц истинности (ТИ).

Статусы сворачиваемых узлов в ПЛС всегда имеют одинаковые значения, так как у них общий родительский узел. Для подграфа, показанного на рис. 9, *а*, первоначальная, максимально возможная область определения найдена исходя из статусов, которые имеют узлы a_1 и a_2. Нулевая (0-строка) и единичная (1-строка) строки вычеркнуты из таблицы истинности ввиду того, что логическая операция $\veebar$ (XOR) разворачивающего узла исключает одновременное присутствие узлов a_1 и a_2 в реализационных графах, равно как и отсутствие любого из них. (Нулевыми и единичными названы строки, которые содержат либо только нули, либо только единицы соответственно.)

Аналогичные рассуждения применены для всех представленных экземпляров сворачиваемых подграфов.

Обратим отдельное внимание на нулевые строки двух таблиц на рис. 9, *а*, и рис. 9, *ж*, помеченные символом √. Своим появлением они обязаны: наличию дополнительного узла a_k, статусу '+' и логическим операциям $\veebar$ или $\vee$ родительского узла. Вследствие этого в части реализационных графов присутствует узел a_k и отсутствуют узлы a_1 и a_2.

Увеличение числа сворачиваемых узлов до трех и более узлов никаким образом не повлияет на сделанные выводы, касающиеся исключения или включения определенных категорий (разновидностей) строк из таблиц истинности, вне зависимости от того, какие это строки: нулевые, единичные или это строки, в которых присутствуют и нули, и единицы. Для каждого экземпляра сворачиваемого подграфа будет характерен свой набор разновидностей строк. (В последующем это свойство будет использовано при разработке алгоритма вычисления статусов сворачивающих соединений.)

5. Экземпляры подграфов комбинированной свертки

Исходя из представленных выше соображений проведем такой же анализ для экземпляров подграфов комбинированной свертки (рис. 10), ограничившись в этом случае заключениями, касающимися только фактов появления (исключения) нулевых и единичных строк в (из) ТИ. (Наличие или отсутствие строк, в которых чередуются нули и единицы, определяется на основании статусов сворачиваемых узлов.) Перечислим сначала отличия, которыми различаются подграфы локальных и комбинированных сверток.

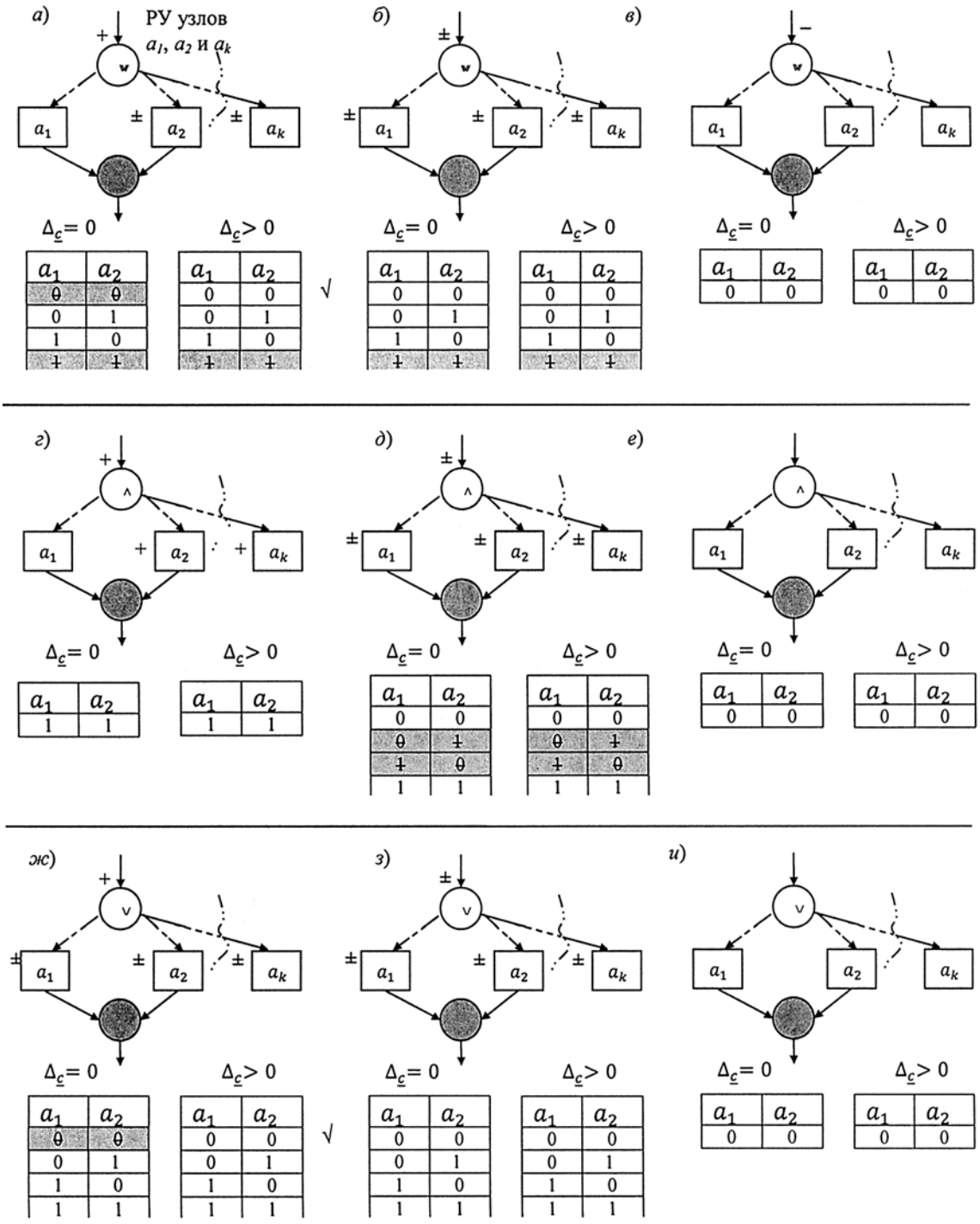


Рис. 9. Свертка узлов, развернутых единственным соединением

Во-первых, при комбинированной свертке ни о каком исчерпывающем влиянии статуса $S_{\overline{\overline{c}}}$ и логической операции $f_{\overline{C}L}(\overline{\overline{c}})$, здесь $\overline{\overline{c}}$ — родительский узел сворачиваемых узлов, на состав $DomF_{\underline{C}}$, как это имеет место в ПЛС, не может быть речи, вследствие того что число $\overline{\overline{c}}$ всегда будет больше единицы (для каждого сворачиваемого узла всегда существует собственный развернувший его узел). В то же время в WF-сети всегда присутствует общий предок, который развернул и сворачиваемые узлы, и узел $\underline{c}$. Роль этого предка играет родительский узел узла $\underline{c}$, который может быть найден в соответствии с изложенным ранее алгоритмом. Его статус и вид его логической операции могут оказывать влияние на одновременное появление или непоявление сворачиваемых узлов в реализационных графах.

Во-вторых, на основании только одного анализа $S_{\overline{\overline{c}}}$ и $f_{\overline{C}L}(\overline{\overline{c}})$, где $\overline{\overline{c}}$ — это уже родительский узел узла $\underline{c}$, сделать однозначное заключе-

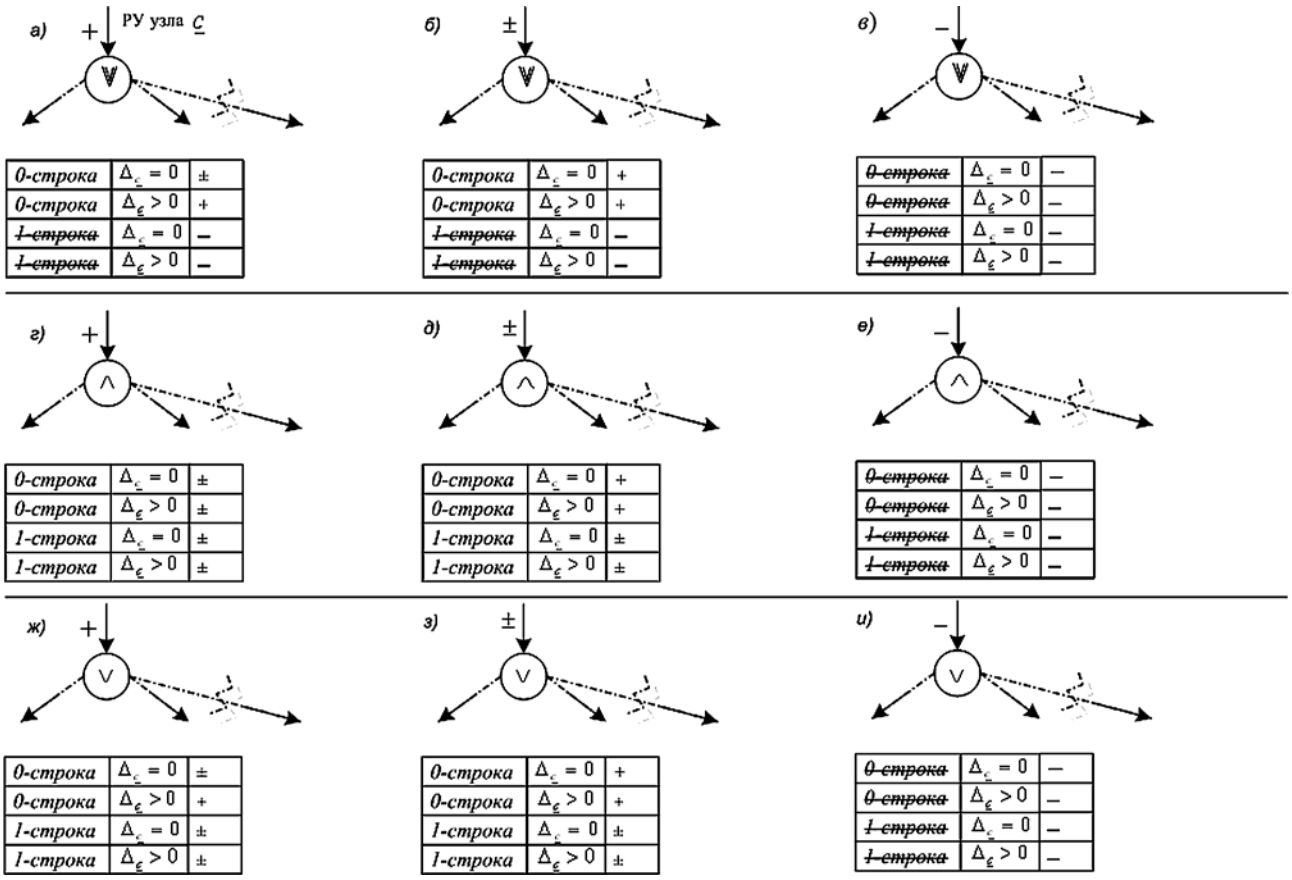


Рис. 10. Свертка узлов, развернутых несколькими соединениями

ние о присутствии или отсутствии 0-строк или 1-строк в ТИ для некоторых экземпляров ПКС нельзя. Можно говорить только о возможности, невозможности или обязательности наличия таких строк.

В-третьих, значение переменной N_U, используемой для нахождении $\Delta_{\underline{c}}$, нельзя вычислить на основании простого суммирования узлов, участвующих в свертке, как это было сделано в случае с ПЛС.

Чтобы показать обязательность, возможность или невозможность появления сворачиваемых узлов в реализационных графах, будем использовать те же символы ‘+’, ‘±’ и ‘–’, что и для значений статусов узлов.

Так, например, для экземпляра ПКС, показанного на рис. 10, *а*, 0-строка может появиться в ТИ только при условии определенных значений статусов сворачиваемых соединений. Действительно, конфигурация данного экземпляра подграфа допускает такую возможность, но ее реализация определится в итоге статусами конкретных сворачиваемых узлов, один из которых, например, может иметь статус ‘+’.

Таким образом, во всех реализациях WF-сетей, если обнаруживаются экземпляры ПКС, в которых допускается возможность одновременного присутствия или отсутствия сворачиваемых узлов (оценка ‘±’ в таблицах истинности на рис. 10), окончательное решение о появлении таких строк принимается на основании статусов сворачиваемых узлов.

Основная сложность, которая возникает при нахождении $DomF_{\underline{C}}$ в ПКС, заключается в определении значения N_U. Для этого требуется найти все исходные (развернутые узлом, являющимся родительским для $\underline{c}$) вершины, из которых в WF-графе есть путь в $\underline{c}$. Сделать это можно в процессе обхода графа на основе алгоритма (который приводится ниже), сохраняя при этом для каждого узла $\underline{c}$ информацию о сворачиваемых им вершинах либо перемещаясь назад по пути из $\underline{c}$ в $\overline{\overline{c}}$ до тех пор, пока не произойдет одно из событий: не будет достигнут $\overline{\overline{c}}$ или посещен какой-либо из узлов $\underline{c}$. Наглядное представление о способе вычисления N_U покажем на примере графа, изображенного на рис. 11. Вершины, сворачиваемые $\underline{c}$-узлами, перечислены в круглых скобках.

Для нахождения вершин, сворачиваемых $\underline{c}_1$, потребуется пройти четыре пути, которые заканчиваются в РУ. Для узла $\underline{c}_2$ — три пути, один из которых завершится в вершине $\underline{c}_1$. Посещение узла $\underline{c}_1$ даст информацию об уже обнаруженных сворачиваемых вершинах a_1 и a_2.

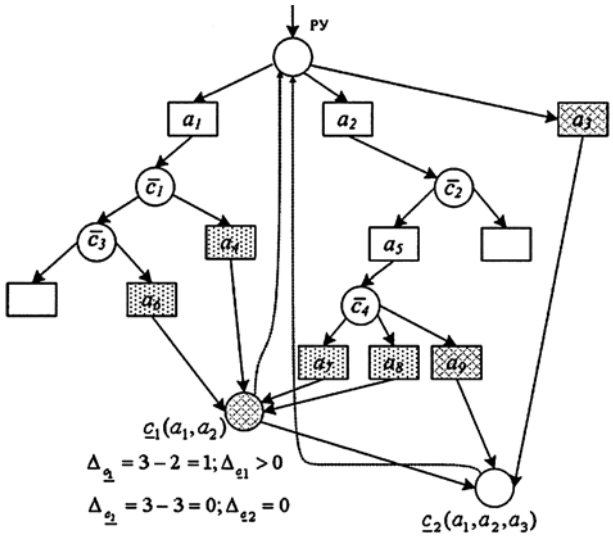
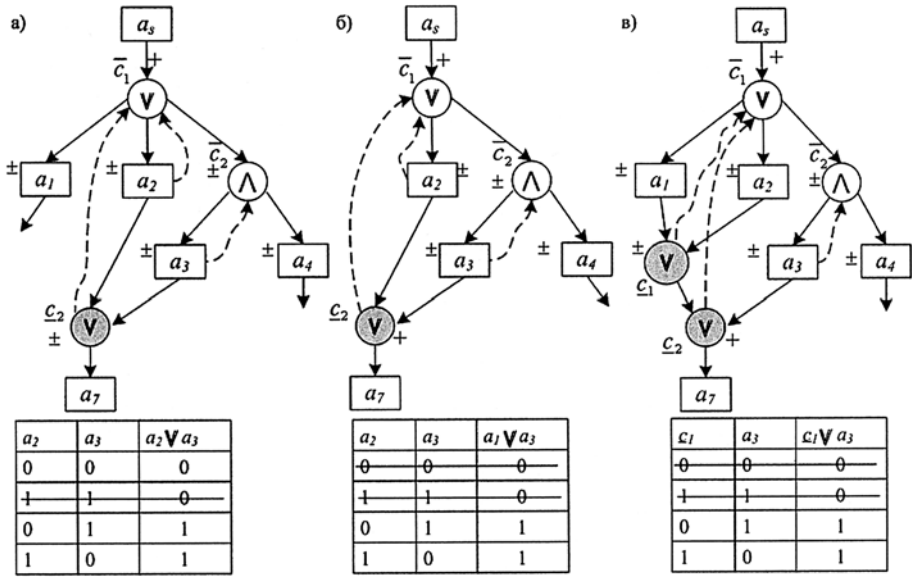


Рис. 11. ПКС с вложенными ПКС

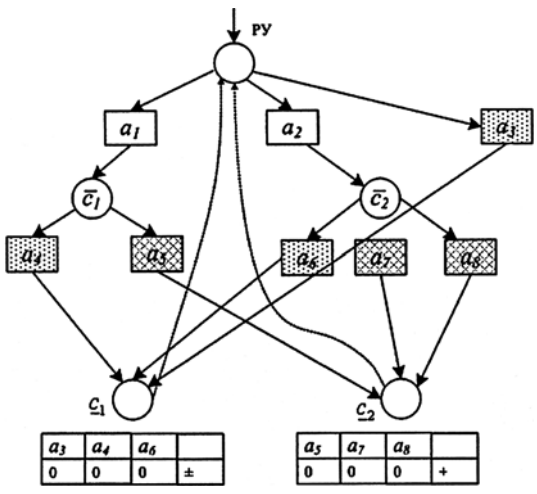


a_2	a_3	$a_2 \vee a_3$
0	0	0
1	1	0
0	1	1
1	0	1

a_2	a_3	$a_1 \vee a_3$
0	0	0
1	1	0
0	1	1
1	0	1

$\underline{c}_1$	a_3	$\underline{c}_1 \vee a_3$
0	0	0
1	1	0
0	1	1
1	0	1

Рис. 12. Области определения функций сворачиваемых соединений



a_3	a_4	a_6	
0	0	0	±

a_5	a_7	a_8	
0	0	0	+

Рис. 13. Простые ПКС с общим РУ

В качестве примера использования полученных результатов приведем еще несколько фрагментов WF-сетей, содержащих ПКС (рис. 12). ТИ на всех рисунках относятся к $\underline{c}_2$.

Ранее в данной работе специальным образом был выделен класс ПКС с вложенными ПЛС. Вычисление статуса сворачивающего соединения ПЛС должно строиться в соответствии с процедурой для ПЛС, дополненной алгоритмом поиска узлов, развернутых узлом, являющимся родительским для $\underline{c}$ анализируемого ПКС.

Завершим анализ сворачиваемых соединений ПКС рассмотрением еще одной конфигурации ПКС, которая не была отдельно классифицирована, воспользовавшись примером (рис. 13).

Подграфы с $\underline{c}_1$ и $\underline{c}_2$ не являются вложенными, поскольку нет пути ни из $\underline{c}_1$ в $\underline{c}_2$, ни из $\underline{c}_2$ в $\underline{c}_1$, в то же время есть в наличии общий родительский узел. В этом случае, как и для всех различных ПКС, отсутствует какое-либо взаимное влияние одного подграфа на другой, способное изменить область определения функций сворачиваемых узлов. Поэтому расчет $S_{\underline{c}_1}$ и $S_{\underline{c}_2}$ необходимо вести как для простых ПКС. Обратим внимание на возможность и обязательность 0-строк в ТИ для $\underline{c}_1$ и $\underline{c}_2$ соответственно применительно к данной конкретной конфигурации WF-сети.

В то же время, в представленном фрагменте сети присутствуют два узла — a_7 и a_8, развернутые одним и тем же узлом $\bar{c}_2$. Следовательно, узел $\underline{c}_2$ одновременно выполняет и локальную, и комбинированную свертки. В связи с этим возникает закономерный вопрос, касающийся процедуры нахождения $Dom\,\underline{c}_2$. Очевидно, что в этом случае $S_{\bar{c}_2}$ и $f_{\bar{C}L}(\bar{c}_2)$ определят возможные сочетания a_7 и a_8 в $Dom\,F_{\underline{c}_2}$.

6. Алгоритм расчета статусов узлов WF-сети

Предложенный метод анализа достижимости узлов WF-сетей уже включает ряд элементов алгоритмического формата, что облегчает его дальнейшую алгоритмизацию и кодирование. В то же время в рамках одной работы изложить организацию, логику и функционирование всей программы, состоящей из нескольких модулей, крайне проблематично. Поэтому сконцентрируемся толь-

ко на ключевых моментах базового (опорного) алгоритма, в соответствии с которым проводятся обход графа и расчет статуса его узлов, а также на вспомогательном алгоритме нумерации вершин графа.

Любой алгоритм бессмысленно строить и анализировать его эффективность, если не принимать в расчет организацию структур, в которых будут размещаться данные, обрабатываемые алгоритмом. Это имеет отношение как к структурам оперативной и долговременной памяти, так и к вспомогательным структурам, которые будут использованы алгоритмом во время расчета.

В качестве структур "постоянного" хранения задействованы реляционные таблицы. Структуры оперативной памяти и вспомогательные структуры представлены двунаправленным списком примыканий и очередью, которая используется для обхода графа (рис. 14).

Двунаправленный список примыканий состоит из базового пятимерного массива длины N, где N — число узлов графа, и двух примыкающих ступенчатых массивов — правостороннего и левостороннего. Каждая строка базового массива содержит сведения об отдельном узле графа, а строки примыкающих массивов — данные о смежных узлах.

Ступенчатый массив представляет собой массив списков. Каждый отдельный список соотносится с n-м узлом графа. В условных правостороннем и левостороннем списках перечисляются вершины, образующие упорядоченные пары с n-м узлом: в правостороннем списке — вершины, исходящие из n, в левостороннем списке — вершины, входящие в n-й узел.

Алгоритм строится "вокруг" очереди, в которую последовательно, в процессе обхода графа, помещаются узлы. На первом шаге из списка узлов выбирается первый узел. Выборка всех последующих узлов проводится на основании информации, содержащейся в правостороннем списке примыканий. Сворачивающие узлы попадают в очередь только в том случае, если посещены все узлы (прошли очередь либо уже присутствуют в ней), являющиеся родительскими по отношению к сворачивающему узлу.

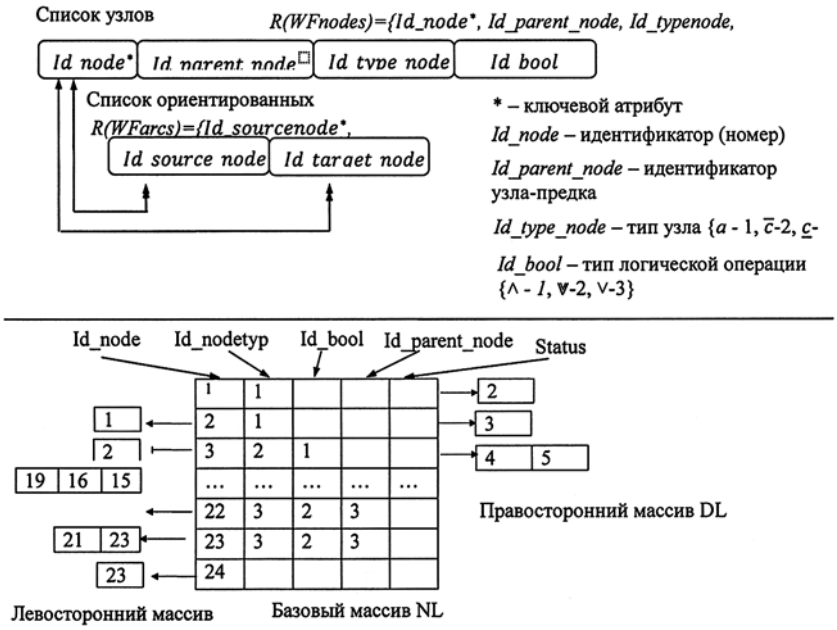


Рис. 14. Структуры для представления WF-сети в оперативной и долговременной памяти

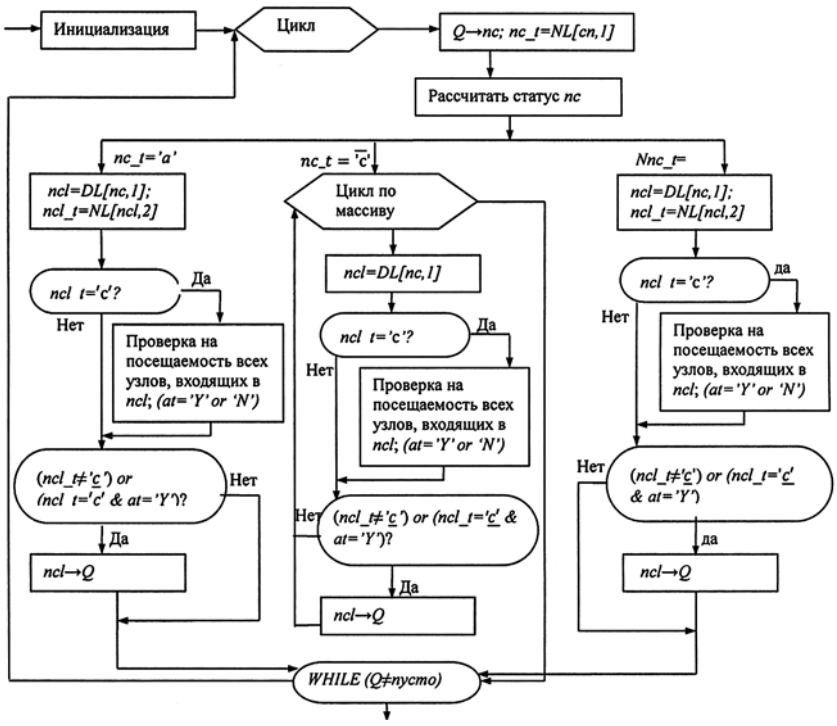


Рис. 15. Блок-схема алгоритма обхода WF-графа

За вычисление статуса узла отвечает метод *Status*(*cn*), принимающий в качестве входного параметра номер узла *cn*. Алгоритм, реализуемый этим методом, предусматривает создание таблиц истинности для сворачиваемых узлов и выполнение инструкций по расчету статуса, которые были представлены в работе ранее.

Блок-схема алгоритма обхода графа и вычисления статусов его узлов показаны на рис. 15.

Алгоритма обхода WF-графа

1. Создать очередь Q, разместить в ней первый элемент из списка узлов $NL[1, 1] \rightarrow Q$ и присвоить ему статус "+": $NL[1, 5] =$ '+'.

2. Организовать цикл с проверкой условия на отсутствие элементов в очереди в конце цикла.

3. Извлечь очередной элемент из очереди $Q \rightarrow nc$ и определить тип извлеченного узла $nc\_t = NL[cn, 2]$.

4. Если $nc\_t =$ *'a'*, то получить прилинкованный к nc узел $ncl = DL[nc, 1]$ и определить его тип $ncl\_t = NL[ncl, 2]$.

4.1. Если $ncl\_t =$ '$\underline{c}$', то проверить, все ли узлы, входящие в *ncl*, посещены. Для этого организовать цикл по строке массива *IL* с индексом *ncl* и удостовериться, все ли узлы, перечисленные в этой строке, получили статусы. Если 'да', то присвоить переменной *at* значение 'Y'.

4.2. Если (($nlt \neq$ '$\underline{c}$') *or* ($nlt =$ '$\underline{c}$ *& at = 'Y'*)), то отправить прилинкованный к nc узел в очередь $ncl \rightarrow Q$.

5. Если $nc\_t =$ '$\overline{c}$', то организовать цикл по строке массива *DL* с индексом nc и получить из *DL*[*nc*, ...] последовательно все узлы, размещая их в *nc_l*.

5.1. Если тип $nc\_l =$ '$\underline{c}$', то выполнить проверку на посещаемость всех входящих узлов в *ncl* узлов (по аналогии с шагом 4.1).

5.2. Если ($nlt \neq$ '$\overline{c}$') *or* ($nt =$ *'a' & nlt =* '$\underline{c}$ *& at = 'Y'*), то отправить прилинкованный к nc узел в очередь $ncl \rightarrow Q$.

6. Если $nc\_t =$ '$\underline{c}$', то организовать цикл по строке массива *DL* с индексом nc и получить последовательно из *DL*[*nc*, ...] все узлы, размещая их в *nc_l*.

6.1. Если тип $nc\_l =$ '$\underline{c}$', то выполнить проверку на посещаемость всех входящих узлов в *ncl* узлов (по аналогии с шагом 4.1).

6.2. Если ($nlt \neq$ '$\overline{c}$') *or* ($nt =$ *'a' & nlt =* '$\underline{c}$ *& at = 'Y'*), то отправить прилинкованный к nc узел в очередь $ncl \rightarrow Q$.

7. Если в очереди не осталось элементов, то завершить алгоритм, если остались — перейти к следующему шагу цикла.

Рассмотрение базового алгоритма завершим текстом программы на языке *C#* (Листинг 1). (Индексация элементов массивов в листинге отличается от индексации в описании алгоритма, поскольку в языке *C* нумерация индексов начинается с нулевого элемента.)

Листинг 1

```
Queue<int> Q = new Queue<int>();                    // Создание экземпляра очереди
int nc, nc_t, ncl, ncl_t; // nc — текущий узел; nc_t — тип текущего узла; ncl — прилинкованный узел;
ncl_t — //тип прилинкованного узла
char at; // признак посещаемости узла
   NL[0, 5] = 1;                                    // Установка статуса исходного узла
   Q.Enqueue(NodesList[0, 0]);                      // Размещение узла в очереди
   do { nc = Q.Dequeue();                           // Извлечение узла из очереди
   if (NL-1, 5] = = 0)
      Status(cn);                                   // Расчет статуса узла
      nc_t = NL[nc − 1, 1]; at = 'Y';
         switch (nc_t) {
            case 1:                                 // Тип узла — a
               ncl = DL[nc − 1][0];
               if (NL[ncl − 1, 1] = = 3) {          // Встретился сворачивающий узел
                        foreach (int j in IL[ncl-1]) {
                                    if (NL[j-1,5] = = 0) { at = 'N';
                                    break; } }
                        if (at = = 'Y') // все родительские узлы посещены
                                    Q.Enqueue(ncl); } else
                                    Q.Enqueue(ncl); break;
               case 2:                              // Тип узла c̄
                foreach (int k in DL[nc − 1]) {
                if (NL[k − 1, 1] = = 3) {           // Встретился сворачивающий узел
                foreach (int j in IL[k − 1]) {
                if (NL[j − 1, 5] = = 0) { at = 'N'; break; }}
                if (at = = 'Y') Q.Enqueue(k); }
                else Q.Enqueue(k) };; break;
```

```
        case 3:                                          // Тип узла c
            ncl = DL[nc − 1][0];
            if (NL[ncl − 1, 1] = = 3) {                  // Встретился сворачивающий узел
            foreach (int j in IL[ncl − 1]) {
            if (NL[j − 1, 5] = = 0) { at = 'N';
            break; }}
            if (at = = 'Y')
            Q.Enqueue(ncl);
            break;
            default: break; }
    } while (Q.Count! = 0);
```

Программа в качестве очереди использует экземпляр стандартного класса *Queue*, входящего в пространство имен *System.Collection*.

Временная сложность алгоритма обхода графа потоков работ может быть установлена на основании числа посещений узлов. Все узлы графа, за исключением сворачивающих узлов, посещаются однократно. Число посещений любого $\underline{c}$ -узла равно числу входящих в него дуг. Отсюда, с учетом несущественного превышения, временная сложность алгоритма составит $O(n + m)$, где n и m — соответственно число узлов и дуг графа. Для сравнения, алгоритм построения дерева достижимости сети Петри требует экспоненциальных объема памяти и времени решения [14].

Представленный алгоритм обхода корректен, если выполняются два условия:

- индекс разворачивающего узла меньше индекса узлов, которые им непосредственно развернуты;
- индекс сворачивающего узла больше индекса узлов, которые он разворачивает.

Выполнение этих двух условий гарантирует, что для элемента, извлекаемого из очереди, перед тем, как будет рассчитан его статус, уже будут известны статусы всех родительских узлов, входящих в сворачивающий подграф. В этой связи обход графа должно предварять выполнение действий, приводящих к нумерации узлов графа, удовлетворяющей представленным условиям.

Поскольку в отличие от процедуры, вычисляющей статусы узлов, алгоритм нумерации носит вспомогательный характер, ограничимся его схематическим представлением, проиллюстрировав его на примере нумерации узлов графа, который будем использовать в качестве контрольного примера (рис. 16).

Требуемый результат может быть получен при использовании процедуры обхода графа по уровням. Стандартный алгоритм обхода гарантирует соблюдение первого условия. Действительно, разворачивающий узел будет всегда посещен раньше узлов, которые он разворачивает, поскольку он всегда располагается на более низком уровне. Что касается выполнения второго условия, то алгоритм обхода следует дополнить блоком, проверяющим обязательность посещения всех узлов, входящих в сворачиваемый узел. Это можно сделать таким же образом, как это сделано в представленном алгоритме обхода WF-графа. Достаточно обратиться к спискам примыкающих узлов, содержащихся в левостороннем массиве примыканий.

Данные о посещаемых узлах и присваиваемых им номерах содержатся в табл. 1. Сворачиваемые узлы в соответствии с алгоритмом могут посещаться несколько раз. И только после последнего посещения им присваивается номер.

В завершение проиллюстрируем работу программы обхода и расчета статусов узлов на примере того же самого графа, что приведен на рис. 16, присвоив $\overline{c}$ - и $\underline{c}$ -узлам определенные

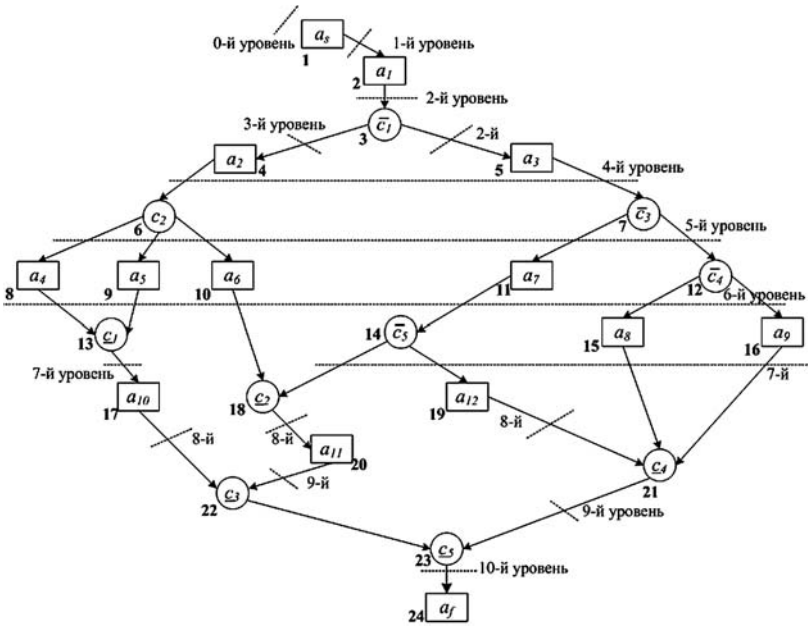


Рис. 16. Нумерация узлов WF-графа

Таблица 1

Обход по уровням и нумерация узлов сети потоков работ

0-й уровень	a_s (1)				
1-й уровень	a_1 (2)				
2-й уровень	$\overline{c}_1$ (3)				
3-й уровень	a_2 (4)	a_3 (5)			
4-й уровень	$\overline{c}_2$ (6)	$\overline{c}_3$ (7)			
5-й уровень	a_4 (8)	a_5 (9)	a_6 (10)	a_7 (11)	$\overline{c}_4$ (12)
6-й уровень	$\underline{c}_1$ (13)	$\underline{c}_2^1$	$\overline{c}_5$ (14)	a_8 (15)	a_9 (16)
7-й уровень	a_{10} (17)	$\underline{c}_2^2$ (18)	a_{12} (19)		
8-й уровень	$\underline{c}_3^1$	a_{11} (20)	$\underline{c}_4$ (21)		
9-й уровень	$\underline{c}_3^2$ (22)	$\underline{c}_5$ (23)			
10-й уровень	a_f (24)				

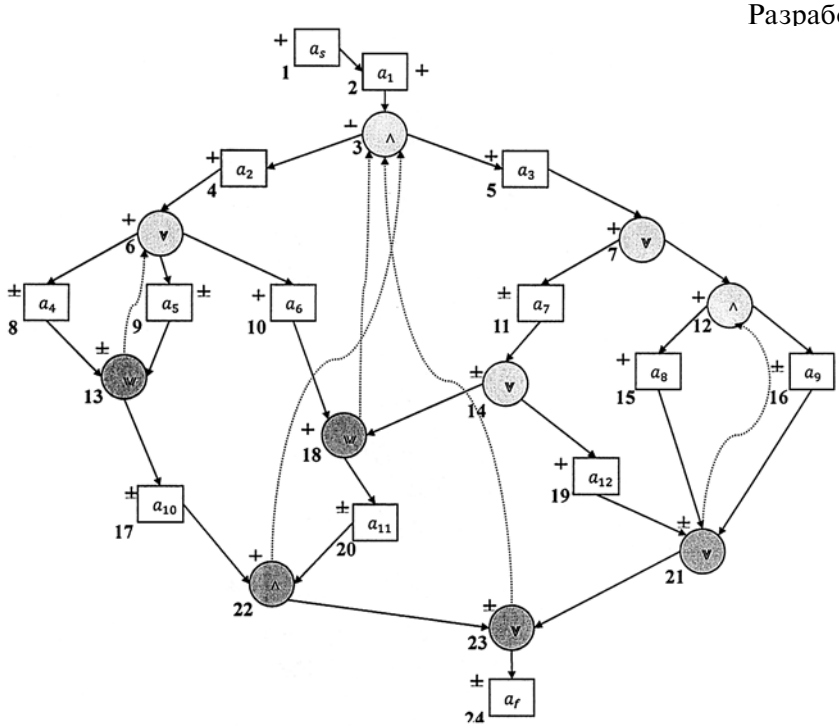


Рис. 17. Статусы узлов WF-графа

логические функции. В результате получим граф, который изображен на рис. 17.

Сделаем это, показав состояние очереди на каждом очередном шаге выполнения тела цикла *Do...While* (табл. 2). Зачеркнутые номера узлов в ячейках таблицы приведены для того, чтобы показать, что сворачиваемые узлы, несмотря на то, что на них ссылаются извлекаемые узлы, не могут быть отправлены в очередь, так как в ней (или за ее пределами) еще остались смежные с $\underline{c}$ узлы.

Выполнение каждого шага завершается извлечением узла из очереди и вычислением его статуса. Извлеченные узлы и их статусы представлены в двух последних строках табл. 2.

Заключение

Разработанный метод определения достижимости узлов WF-сетей дает возможность отказаться от использования традиционных методов, предусматривающих преобразование анализируемых моделей потоков работ в сети Петри. В основе метода анализ разворачивающих и сворачиваемых соединений на предмет их достижимости или недостижимости, что может иметь место для отдельных экземпляров сети. Для этого используются булевы функции, соответствующие логическим операциям разворачивающих и сворачивающих узлов. Установлены факторы, отражающие структурные особенности организации сети, и исследован механизм их влияния на содержание области определения булевых функций.

Реализация процесса с помощью разработанного метода предусматривает однократное посещение вершин графа с одновременным расчетом их

Таблица 2

Состояние очереди в процессе обхода WF-сети

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
1	2	$\overline{3}$	5	$\overline{6}$	$\overline{7}$	10	$\overline{12}$	~~13~~	13	~~18~~	$\overline{14}$	16	17	19	~~21~~	~~21~~	~~22~~	20	22	21	~~23~~	23	24
			4	5	6	9	11	$\overline{12}$	$\overline{12}$	13	13	15	16	18	19	19	19	19	20	22	21		
						8	10	11	11	$\overline{12}$	$\overline{12}$	$\overline{14}$	15	17	18	18	18						
						$\overline{7}$	9	10	10	11		13	$\overline{14}$	16	17	17							
							8	9						15	16								
1	2	3	4	5	6	$\overline{7}$	8	9	10	11	$\overline{12}$	13	$\overline{14}$	15	16	17	18	19	20	22	21	23	24
+	+	+	+	+	+	+	±	±	±	±	±	±	±	±	±	±	±	±	±	±	±	±	±

статуса в соответствии с предложенным оригинальным алгоритмом обхода WF-сети. Алгоритм использует двунаправленные списки примыканий для хранения информации о структуре графа и очередь в качестве структуры для размещения узлов, посещаемых в процессе обхода.

Опираясь на полученные результаты, можно ставить задачи и развивать методы анализа сетей потоков работ, бездефектность которых предполагает достижение не только конечной точки процесса, а некоторого, наперед заданного, подмножества узлов, расположенных в произвольных точках сети, равно как и совокупности таких подмножеств, моделирующих возможные альтернативы в реализации процессов.

Список литературы

1. **Ван дер Аалст В., ванн Хей К.** Управление потоками работ: модели, методы и системы. М.: ФИЗМАТЛИТ, 2007. 316 с.
2. **Van der Aalst W. M. P.** Formalization and verification of event-driven process chains // Information and software technology. 1999. Vol. 41 (10). P. 639—650.
3. **Van der Aalst W. M. P., van Hee K. M., ter Hofstede A. H. M., Sidorova N., Verbeek H. M. W., Voorhoeve M., Wynn M. T.** Soundness of workflow nets: classification, decidability, and analysis // Formal aspects of computing. 2011. Vol. 23 (3). P. 333—363.
4. **Verbeek H. M. W., van der Aalst W. M., ter Hofstede A. H. M.** Verifying workflows with cancellation regions and OR-joins: An approach based on relaxed soundness and invariants // The computer journal. 2007. Vol. 50 (3). P. 294—314.
5. **Kindler E.** On the semantics of EPCs: A framework for resolving the vicious circle // Data and knowledge engineering. 2006. Vol. 56 (1). P. 23—40.
6. **Wynn M. T., van der Aalst W. M. P., ter Hofstede A. H. M., Edmond D.** Verifying workflows with cancellation regions and OR-joins: An approach based on reset nets and reachability analysis // Proc. of *International Conference on Business Process Management (BPM 2006)*. Vol. 4102. *Lecture Notes in Computer Science*. Springer-Verlag, Berlin, 2006. P. 389—394.
7. **Boucheneb H., Barkaoui K.** Partial order reduction for checking soundness of time workflow nets // Information sciences. 2014. Vol. 282. P. 261—276.
8. **Clempner J.** Verifying soundness of business processes: A decision process Petri nets approach // Expert Systems with Applications. 2014. Vol. 41. P. 5030—2040.
9. **Башкин В. А., Ломазова И. А.** О разрешимости бездефектности для сетей потоков работ с неограниченным ресурсом // Моделирование и анализ информационных систем. 2013. Т. 20 (4). С. 23—40.
10. **Cravo G.** Applications of propositional logic to workflow analysis // Applied mathematics letters. 2010. Vol. 23. P. 272—276.
11. **Родионов А. Н.** Некоторые синтаксические и семантические ограничения сценарных моделей бизнес-процессов // Информационные технологии. 2017. Т. 23, № 4. С. 273—281.
12. **Bartak R., Rovensky V.** On verification of nested workflows with extra constraints: From theory to practice // Expert systems with applications. 2014. Vol. 41, is. 3. P. 904—918.
13. **Sidorova N., Stahl C., Trcka N.** Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible // Information system. 2011. Vol. 36. P. 1026—1043.
14. **Cheng A., Esparza J., Palsberg J.** Complexity results for 1-safe nets // Theoretical computer science. 1995. Vol. 147, is. 1—2. P. 117—136.

DOI: 10.17587/it.24.274-288

A. N. Rodionov, D. Sci., Leading Researcher, ran@newmail.ru
Computer Centre Of Far-Eastern Branch of RAS, Khabarovsk, Russia

Reachability of Workflow Nodes: The Method of Structural Analysis and Algorithm

A workflow net is a sort of directed graphs that simultaneously represent a set of ordinary graphs by means of inserting logical nodes. Such net adequately simulates any business processes configurations.

This article deals with a standard workflow model that composes of functional and logical nodes and satisfies several limitations with respect to net organization.

Gist of verifying workflow net soundness is to ascertain a fact of the whole or partial reachability or inaccessibility of activities, which are functional nodes. This issue is of immediate interest for all realms of process-aware information systems engineering. Our technique uses propositional logic and some aspects of graph theory to analyze reachability of every join and split connectors along with activities nodes. We unveil some structural configurations of a net that effect appearance or failure of activity nodes at all net instances. As a consequence a reachability issue result in a task of graph traversal and calculation of vertexes statuses. A status testifies whether a vertex can be reach or not at the whole or separate net instances. To status estimate we use Boolean functions, which meet logical operations that assign to logical nodes, with domains that are calculated on the basis of net structural parameters values. Finally, we resort to the original threading of a workflow net to improve the traversal and reachability algorithm efficiency.

Keywords: *business process, workflow, verification, soundness, reachability analysis*

References

1. **Van der Aalst V., Van Hej K.** *Upravlenie potokami rabot: modeli, metody i sistemy* (Wil van der Aalst, Kees van Hee Workflow management: models, methods, and systems), Moscow, FIZMATLIT, 2007. 316 p.

2. **Van der Aalst W. M. P.** Formalization and verification of event-driven process chains, *Information and software technology*, 1999, vol. 41 (10), pp. 639—650.

3. **Van der Aalst W. M. P., van Hee K. M., ter Hofstede A. H. M., Sidorova N., Verbeek H. M. W., Voorhoeve M., Wynn M. T.** Soundness of workflow nets: classification, decidability, and analysis, *Formal aspects of computing*, 2011, vol. 23 (3), pp. 333—363.

4. **Verbeek H. M. W., van der Aalst W. M., ter Hofstede A. H. M.** Verifying workflows with cancellation regions and OR-joins: An approach based on relaxed soundness and invariants, *The computer journal*, 2007, vol. 50 (3), pp. 294—314.

5. **Kindler E.** On the semantics of EPCs: A framework for resolving the vicious circle, *Data and knowledge engineering*, 2006, vol. 56 (1), pp. 23—40.

6. **Wynn M. T., van der Aalst W. M. P., ter Hofstede A. H. M., Edmond D.** Verifying workflows with cancellation regions and OR-joins: An approach based on reset nets and reachability analysis, *International Conference on Business Process Management (BPM 2006), vol. 4102, of Lecture Notes in Computer Science*, Springer-Verlag, Berlin, 2006, pp. 389—394.

7. **Boucheneb H., Barkaoui K.** Partial order reduction for checking soundness of time workflow nets, *Information sciences*, 2014, vol. 282, pp. 261—276.

8. **Clempner J.** Verifying soundness of business processes: A decision process Petri nets approach, *Expert Systems with Applications*, 2014, vol. 41, pp. 5030—2040.

9. **Bashkin V. A., Lomazova I. A.** O razreshimosti bezdefektnosti dlja setej potokov rabot s neogranichennym resursom, *Modelirovanie i analiz informacionnyh sistem*, 2013, vol. 20 (4), pp. 23—40. (**Bashkin V. A., Lomazova I. A.** On the Decidability of Soundness of Workflow Nets with an Unbounded Resource, Modeling and Analysis of Information Systems, 2013, vol. 20 (4), pp. 23—40.) (in Russian).

10. **Cravo G.** Applications of propositional logic to workflow analysis, *Applied mathematics letters*, 2010, vol. 23, pp. 272—276.

11. **Rodionov A. N.** *Nekotorye sintaksicheskie i semanticheskie ogranichenija scenarnyh modelej biznes-processov* (Some syntactic and semantic limits on scenary business process models), *Informacionnye tehnologii*, 2017, vol. 23, no. 4, pp. 273—281 (in Russian).

12. **Bartak R., Rovensky V.** On verification of nested workflows with extra constraints: From theory to practice, *Expert systems with applications*, 2014, vol. 41, is. 3, pp. 904—918.

13. **Sidorova N., Stahl C., Trcka N.** Soundness verification for conceptual workflow nets with data: Early detection of errors with the most precision possible, *Information System*, 2011, vol. 36, pp. 1026—1043.

14. **Cheng A., Esparza J., Palsberg J.** Complexity results for 1-safe nets, *Theoretical Computer Science*, 1995, vol. 147, is. 1—2, pp. 117—136.

Адрес редакции:

107076, Москва, Стромынский пер., 4

Телефон редакции журнала **(499) 269-5510**
E-mail: it@novtex.ru

Технический редактор *Е. В. Конова*.
Корректор *З. В. Наумова*.
Сдано в набор 09.02.2018. Подписано в печать 27.03.2018. Формат 60×88 1/8. Бумага офсетная.
Усл. печ. л. 8,86. Заказ IT418. Цена договорная.

Журнал зарегистрирован в Министерстве Российской Федерации по делам печати, телерадиовещания и средств массовых коммуникаций.
Свидетельство о регистрации ПИ № 77-15565 от 02 июня 2003 г.

Оригинал-макет ООО "Адвансед солюшнз". Отпечатано в ООО "Адвансед солюшнз".
119071, г. Москва, Ленинский пр-т, д. 19, стр. 1.