ИСПОЛЬЗОВАНИЕ СИСТЕМЫ АВТОМАТИЧЕСКОЙ ВЕРИФИКАЦИИ И ОПТИМИЗАЦИИ ПОТОКОВ РАБОТ ДЛЯ АНАЛИЗА ПРОЦЕССОВ ОБРАБОТКИ ИНФОРМАЦИОННЫХ РЕСУРСОВ В ЭЛЕКТРОННОЙ БИБЛИОТЕКЕ

[1]Каленкова А.А.

[1]Учреждение Российской академии наук Вычислительный центр им. А.А.Дородницына РАН

119333, Москва, ул. Вавилова, 40

akalenkova@ultimeta.ru

Введение

Процессы обработки информационных ресурсов в электронных библиотеках, как и все производственные процессы, состоящие из подзадач, которые выполняются различными участниками и программными приложениями, нуждаются в строгой формализации и автоматизации исполнения. Широкое распространение имеет теория потоков работ: потоком работ принято считать формальное описание процедуры передачи данных и управления между участниками процесса в соответствии с определенными правилами его выполнения, системы управления потоками работ в свою очередь решают задачи определения и исполнения потоков работ [1].

Единый управляющий механизм и строгая формализация описания производственного процесса на некотором языке обеспечивают:

- координацию потока управления и потока данных;
- полную автоматизацию отдельных участков процесса (появляется возможность взаимодействия с «программными» исполнителями заданий в рамках некоторого унифицированного интерфейса);
- модульность процесса, повторное использование описаний;
- хорошую управляемость процессом, доступ к данным любого активного этапа;
- четкое ролевое разделение участников процесса;
- возможность сбора статистики выполнения процесса для последующего анализа;
- а также возможность применения унифицированных алгоритмов оптимизации процесса.

Известны системы управления потоками работ с открытым кодом, позволяющие интегрировать как программные, так и человеческие ресурсы [2, 3], эффективно решающие указанные задачи.

В настоящее время определенное внимание уделяется алгоритмам автоматического анализа потоков работ – их верификации [4-7] и оптимизации [8, 9]. Эти алгоритмы особенно необходимы там, где процессы имеют объемную и запутанную структуру. Так, например, процессы обработки информационных ресурсов в электронных библиотеках достаточно сложны, так как в них участвуют несколько организаций, между которыми идет интенсивный обмен данными и возможны ветвления. Нами был предложен новый алгоритм верификации потоков работ [10], имеющий полиномиальную вычислительную сложность. В ходе работы алгоритма верификации определяются условия выполнения атомарных действий, что позволило также создать алгоритм оптимизации потоков работ по времени выполнения – алгоритм автоматического распараллеливания действий, независящих по данным. В отличие от известных алгоритмов структурной оптимизации потоков работ, базирующихся на применении шаблонных преобразований [8, 9], предлагаемый алгоритм оптимизации распараллеливает поток работ, содержащий произвольные ветвящиеся

структуры. Эти алгоритмы были реализованы в Системе автоматической верификации и оптимизации потоков работ [11]. Теперь целью нашей работы является формализация и анализ процессов обработки информационных ресурсов в рамках ЭБ «Научное наследие России» [12].

Языки описания потоков работ и системы управления потоками работ

Язык описания потоков работ должен быть достаточно полным так, чтобы аналитик производственного процесса мог указать, какие действия необходимо выполнить для получения заданного результата, какова последовательность их выполнения, кто выполняет действия, каковы входные/выходные параметры действий и самого процесса, какие ресурсы необходимы для выполнения каждого действия. В настоящее время наиболее распространенным графическим стандартом описания потоков работ является нотация BPMN (Business Process Management Notation)[13], которая впервые была предложена в 2003 году, она включила в себя все основные конструкции известных ранее графических стандартов описания потоков работ. Графическое представление производственного процесса имеет свои преимущества: оно понятно всем участникам процесса, также по графическому описанию может быть автоматически построен исполняемый поток работ. Нотация BPMN 2.0 сама включает сущности и их свойства, позволяющие программным и человеческим ресурсам участвовать в исполнении формализованного процесса в некоторой среде интерпретации BPMN 2.0. Среди исполняемых XML-языков описания потоков работ можно выделить два наиболее известных – это XPDL (XML Process definition language) [14] и BPEL (Business Process Execution Language) [15]. Язык XPDL является графовым: задания и маршрутизаторы – это вершины графа, а поток управления представлен дугами. В качестве атомарных действий XPDL могут быть указаны как программные модули, так и задания, исполняемые участниками процесса. BPEL – это блочно-графовый язык: в нем структура потока работ описывается вложенными блоками, однако некоторые действия могут связаны дополнительными потоками управления. BPEL-процесс представляет собой композицию веб-сервисов и полностью автоматизирован. В 2007 году был предложен стандарт BPEL4People [16], который является расширением BPEL и позволяет указывать задания, исполняемые участниками процесса.

Несмотря на существование различных языков описания потоков работ, системы управления потоками работ имеют общую архитектуру (рис. 1) [1].

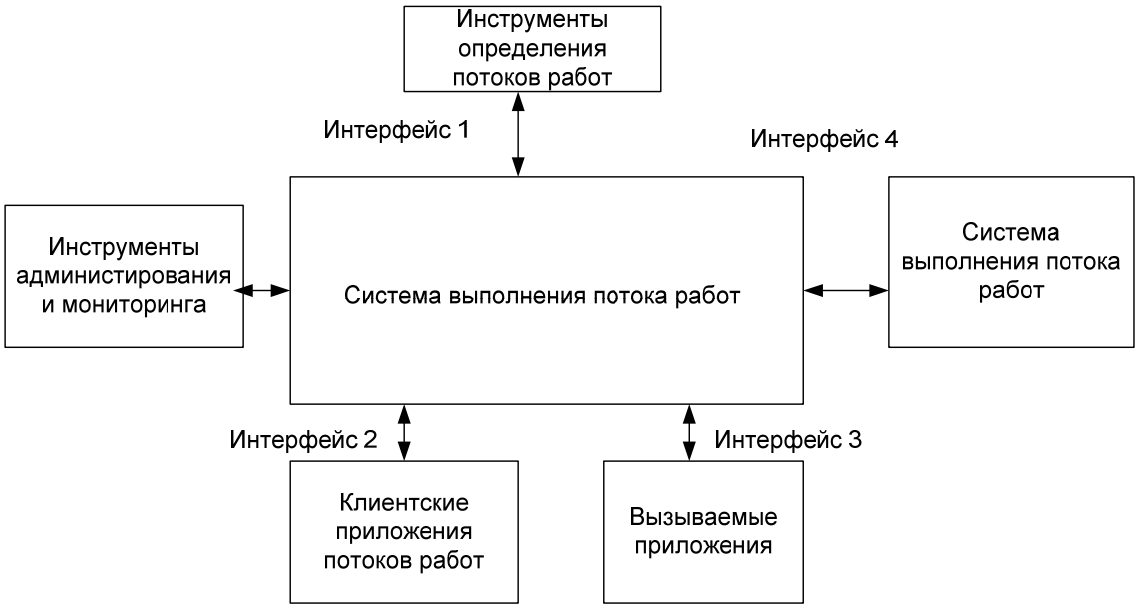

Рис. 1. Архитектура системы управления потоками работ

Формальный язык описания потоков работ

Существуют различные языки описания потоков работ, и они имеют достаточно сложную формально определенную семантику. Тем не менее, в их основе лежит один и тот же базис, определяемый набором шаблонов [17], которые были сформированы практикой использования потоков работ для формализации производственных процессов. Нами был выделен базисный набор конструкций, позволяющий реализовывать основные шаблоны потока управления [17], и на его основе разработан специальный Формальный язык описания потоков работ [18]. Этот язык дает возможность создавать универсальные алгоритмы анализа потоков работ без привязки к конкретному языку. Рассмотрим его подробнее:

Поток работ задается формально как шестерка $WF = (\Sigma, \Omega, A_0, \Phi, \Theta, \Psi)$.

1. Множество Σ - это множество всех локальных генерируемых в процессе выполнения потока работ событий. Информация о произошедших локальных событиях доступна только экземпляру потока работ, который сгенерировал эти события.
2. Множество действий $\Omega = \Omega_{comp} \cup \Omega_{simp}$ состоит из множества составных и простых действий.
3. Начальное действие $A_0 \in \Omega$. Действие, с которого начинается выполнение потока работ.
4. Множество строковых переменных, определенных уровне экземпляра потока работ Φ. В любой момент времени переменной ставится в соответствие некоторое значение. Если при выполнении потока работ переменной не присвоено никакого значения, считается, что ее значение – произвольная строка символов.
5. Множество предикатов Θ. Предикаты являются предопределенным функциями, принимающими значение булевского типа, и использующимися в логических выражениях условных конструкций потока работ.
6. Множество формул Ψ, описывающих структуру действий потока работ.
   Каждая из формул этого множества определяет структуру составного действия, идентификатор которого указан в ее левой части и имеет вид: $A =< activity >$, где A - это имя составного действия.

Опишем синтаксис формул потока работ с помощью формы Бэкуса – Наура. Действия потока работ могут быть простыми и составными: $< activity >::=< simpleActivity > | < compositeActivity >$. Составное действие строится из действий потока работ с использованием следующих основных правил активации вложенных действий (правила расположены в порядке убывания их приоритета).

- Параллельная активация - '&'. Активация действий происходит в асинхронном режиме: порождается набор параллельных процессов. При этом породивший их процесс (корневой) продолжает свое выполнение. Использование правила параллельной активации для единичного действия подразумевает его асинхронную активацию параллельно корневому процессу.
- Последовательная активация - '.'.

Или составное действие является специальным составным действием.

- Условный выбор IF. Проверяется логическое выражение, построенное на множестве предикатов с использованием логических операторов '&' (конъюнкция), '|'

(дизъюнкция), '!' (отрицание), а также скобок для расстановки приоритетов. Переменные задаются своими именами. Если значение выражения – '*истина*'('*true*'), то запускается первое действие (второй аргумент), иначе – второе (третий аргумент).

- Цикл *WHILE* . Аналогично условному выбору вычисляется логическое выражение, пока значение выражения - '*истина*'('*true*') выполняется указанное действие.

$< compositeActivity >::= ['('][< activity >] < activationRule >< activity > [')']$

$|'IF('< booleanExpression >','< activity >','< activity >')'$

$|'WHILE('< booleanExpression >','< activity >')'$

$< activationRule >::= '.'|'\&'$

В соответствии с приведенным синтаксисом поток работ представлен как композиция составных действий. В формулах потока работ для простых действий указываются входные (имена переменных или константы) и выходные параметры (имена переменных). К простым действиям также относятся несколько специальных системных действий:

- Генерация события с заданным именем - $CREATE(x)$, $x \in \Sigma$.
- Ожидание события с одновременным удалением информации о нем, действие атомарно - $\overline{WAIT}(x)$, $x \in \Sigma$.
- Пустое действие, не выполняющее никакой работы, - $EMPTY$.

$< simpleActivity >::= ['<'< \text{var}iables / cons\,\text{tan}ts'>'] < activityName > ['<'< \text{var}iables >'>']$

$|'CREATE('< identifier >')'|'\overline{WAIT}('< identifier >')'|'EMPTY'$

Декомпозиция потока работ на составные действия, описываемые отдельными формулами, не определяет дополнительной функциональной характеристики потока работ. Она служит только для более наглядного представления потока работ.

Для пояснения формального описания потока работ приведем пример. Предлагаемый поток работ включает в себя три действия. Два из них A и B выполняются параллельно. Последнее действие C выполняется после синхронизации двух параллельных ветвей процесса. В примере введены дополнительные действия A', B', C', которые определяют «полезную» работу.

$WF = (\Sigma, \Omega, A_0, \Phi, \Theta, \Psi)$

$\Sigma = \{a, b\}$

$\Omega = \{A_0, A, B, C, (A`, 0, 1), (B`, 0, 1), (C`, 2, 1)\}$

$\Phi = \{x, y, z\}$

$\Theta = \varnothing$

$\Psi = \{A_0 = A \,\&\, B \,\&\, C, A = A' < x > .CREATE(a), B = B' < y > .CREATE(b),$

$C = WAIT(a).WAIT(b). < x, y > C' < z >\}$

Предлагаемый язык – это упрощенный вариант формального языка описания потоков работ, который предложен нами в работе [18]. Это упрощение связано с тем, что в ходе верификации и структурной оптимизации потока работ нас главным образом интересует поток управления и зависимости по данным между простыми действиями, поэтому мы рассматриваем только глобальные переменные уровня потока работ, имеющие строковый тип. Кроме того, мы исследуем исполнение отдельного экземпляра потока работ вне зависимости от среды (работы других экземпляров потоков работ).

Система автоматической верификации и оптимизации потоков работ

Для анализа конструкций потока работ, заданного с помощью Формального языка описания потоков работ, необходимо представить поток работ в виде направленного графа. Нами была предложена соответствующая модель – Размеченные графы анализа потоков работ, которые могут быть построены по формальному описанию потока работ в

соответствии с алгоритмом [18]. Дадим определения Графа анализа потока работ и Размеченного графа анализа потока работ.

Определение : Графом анализа потока работ (или просто графом анализа) называется шестерка $AG = (N, T, t_s, C, M, F)$:

- N - конечное множество вершин;
- $T \subseteq N$ - конечное множество заданий;
- $t_s \in T$ - начальное задание;
- $C \subseteq N$ - конечное множество вершин выбора;
- $M \subseteq N$ - конечное множество вершин слияния;
- $F \subseteq N \times N$ - поток управления.

При этом выполнены условия:

- у начального задания нет входящих дуг;
- для каждой вершины $n \in N$ существует путь из t_s в n;
- у заданий, в отличие от других вершин, может не быть исходящих потоков управления;
- каждая вершина выбора имеет по крайней мере две исходящие дуги.

Граф анализа потока работ – это направленный граф, в котором вершины делятся на задания, выполняющие полезную работу, и координаторы выбора или слияния (рис. 2).

Как и сеть Петри (Petri net) [4], граф анализа обладает не только статической, но и динамической составляющей – дуги графа могут содержать некоторое количество фишек (tokens) или не содержать их вовсе. В отличие от сетей Петри фишки располагаются не в вершинах графа, а на дугах. Перед началом исполнения потока работ исходящая дуга начального задания содержит одну фишку, остальные дуги графа не содержат фишек.

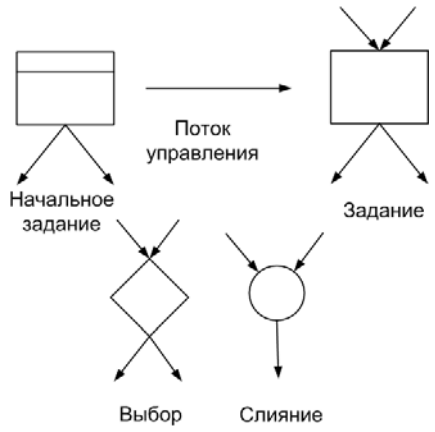

Рис. 2. Элементы Графа анализа потока работ

Задание одновременно синхронизирует и разветвляет поток управления: ожидается появление фишек на всех входящих дугах, по одной фишке забирается с каждой дуги, далее выполняется задание и добавляется фишка на каждую исходящую дугу. При выборе ожидается появление фишек на всех входящих дугах, по одной фишке забирается с каждой входящий дуги, и добавляется по фишке на некоторые из исходящих дуг. И, наконец, при слиянии забирается фишка с одной из входящих дуг (если фишки есть на нескольких дугах, выбор недетерминирован) и передается фишка на исходящую дугу. Исполнение потока работ завершается, когда ни одна вершина не может быть активирована.

Введем ряд обозначений: *Id* - это множество идентификаторов, *Str* - множество строк, $Boolean = \{true, false\}$, знак $\rightharpoonup$ задает частичную функцию. Для символов, формирующих идентификаторы и строки, допускается использование различного набора

индексов. Через V_{in} и V_{out} обозначим множества векторов произвольной конечной длины с координатами из множества Id и введем понятие размеченного графа анализа.

Определение: Размеченным графом анализа потока работ (или просто размеченным графом анализа) называется кортеж $LAG = (N, T, t_s, C, M, F, L_t, L_{in}, L_{out}, L_{ch}, L_{cond}, L_f)$:

- (N, T, t_s, C, M, F) - граф анализа;
- $L_t \in (T \setminus \{t_s\} \to Id)$ - каждому заданию (за исключением начального задания) ставится в соответствие его название;
- $L_{in} \in (T \setminus \{t_s\} \cup C \to V_{in})$ - некоторым заданиями и некоторым вершинам выбора ставится в соответствие список входных параметров;
- $L_{out} \in (T \setminus \{t_s\} \to V_{out})$ - некоторым заданиями ставится в соответствие список выходных параметров;
- $L_{ch} \in (C \to Str)$ - каждой вершине выбора ставится в соответствие условие выбора;
- $L_{cond} \in (F' \to Boolean)$, $F' \subseteq F$, $f' = (n_1, n_2) \in F' \Leftrightarrow n_1 \in C$ - всем потокам управления, исходящим из вершин выбора, ставится в соответствие условие,

  при этом $\forall c \in C \; \exists f_1, f_2 : L_{cond}(f_1) = true, L_{cond}(f_2) = false$, то есть из каждой вершины выбора исходят дуги, как с пометкой $L_{cond} = true$, так и с пометкой $L_{cond} = false$;
- $Lf \in (F \to \mathrm{Id} \cup \{'\&'\})$ - некоторым потокам управления ставится в соответствие идентификатор или амперсанд.

Фактически размеченный граф анализа – это граф анализа, дополненный некоторыми пометками. Семантика размеченного графа анализа отличается от семантики графа анализа тем, что вершина выбора передает фишки на все дуги с пометкой $L_{cond} = true$ или на все дуги с пометкой $L_{cond} = false$. Размеченный граф анализа позволяет указать в качестве входных и выходных параметров глобальные переменные потока работ и дает возможность анализировать зависимости по данным.

Система автоматической верификации и оптимизации потоков работ (Система) позволяет создавать и проверять на соответствие определению, Размеченные графы анализа потоков работ (рис. 3).

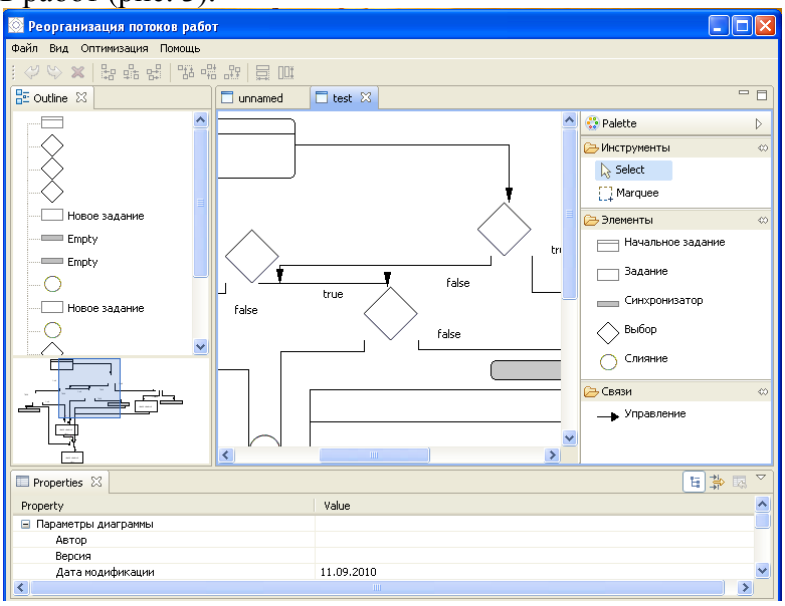

Рис. 3. Система автоматической верификации и оптимизации потоков работ

В настоящее время ведется работа по реализации алгоритмов выгрузки и загрузки потоков работ, заданных на Формальном языке описания потоков работ, в Систему. Проанализировано и реализовано отображение между BPEL и Формальным языком описания потоков работ, на основе которого несложно будет построить отображение между BPEL4People и Формальным языком описания потоков работ. Таким образом, планируется использовать Систему автоматической верификации и оптимизации потоков работ для анализа BPEL4People процессов обработки информационных ресурсов в ЭБ (рис. 4).

Рис. 4. Схема автоматической верификации и оптимизации BPEL4People процесса

Алгоритм автоматической верификации, реализованный в Системе, позволяет работать с вложенными циклами и произвольными перекрывающимися структурами Размеченных графов анализа потоков работ на каждом уровне вложенности. Выделение уровней вложенности построено на основе известного алгоритма проверки графа на сводимость [19]. При этом дополнительно проверяется, что не существует двух или более обратных дуг, имеющих общий конец и нет циклов, обладающих, по крайней мере, двумя выходами (двумя вершинами, соединяющими тело цикла с внешними по отношению к циклу вершинами). Для каждого выделенного цикла проверяются вершина инициирующая цикл и вершина выхода из цикла. Если вершиной инициирующей цикл, является задание (рис.5а) или вершина выбора (рис.5б), то обратная дуга не активируется, и будет образован «мертвый цикл». Если же цикл инициирует вершина слияния (рис.5в), то конфликта не возникнет.

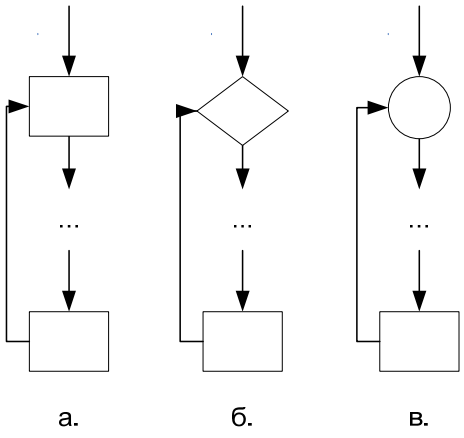

Рис. 5 Анализ вершины инициирующей цикл

Вершина слияния не может быть вершиной выхода из цикла, так как имеет единственный исходящий поток управления. Если вершиной выхода из цикла является

задание (рис.6а), то возникнет «бесконечный цикл», при этом исходящий поток управления этого задания будет активирован бесконечное число раз. Если вершиной выхода из цикла является вершина выбора (рис.6б), то необходимо, чтобы все исходящие дуги, соединяющие ее с внешними по отношению к телу цикла вершинами, имели одинаковые значения пометок L_{cond}, при невыполнении этого условия образуется так называемый «ветвящийся цикл». Если же существует исходящая дуга вершины выхода из цикла, не соединяющая эту вершину с внешними вершинами, которая имеет значение пометки L_{cond} равное значениям пометок L_{cond} этих дуг, то возникнет «бесконечный цикл».

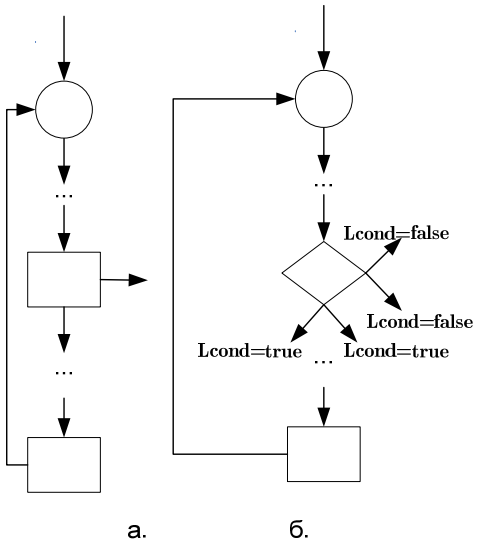

Рис. 6 Анализ вершины выхода из цикла

Верификация ациклического графа позволяет выявлять несбалансированное использование маршрутизирующих элементов. Конфликт типа «тупик» (deadlock) возникает в том случае, если лишь некоторые из входящих дуг синхронизирующей вершины активированы, и синхронизирующая вершина будет бесконечно долго ожидать активации. Конфликт типа «недостаток синхронизации» (lack of synchronization) произойдет, если вершина слияния была активирована несколько раз вследствие активации различных входящих дуг.

Поставим каждой вершине выбора в соответствие уникальный идентификатор. Построим алгоритм определения условий выполнения вершин и активации дуг. В ходе этого алгоритма будет проведен поиск в ширину и для каждой вершины и для каждой дуги будет определено условие выполнения и условие активации соответственно, являющиеся булевыми функциями над множеством идентификаторов вершин выбора. Условие выполнения начальной вершины положим равным $true$. Условие активации исходящей дуги вершины выбора формируется как конъюнкция условия выполнения вершины выбора и ее идентификатора или его отрицания в зависимости от пометки L_{cond}. Условие выполнения синхронизирующей вершины определяется как конъюнкция условий активации входящих дуг (рис. 7).

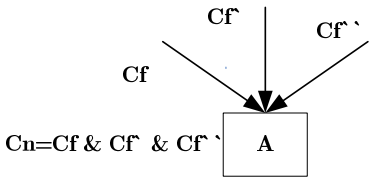

Рисунок 7. Условие активации синхронизирующей вершины

Если не выполнено условие $Cf = Cf` = Cf``$, возможен случай, когда будет активирована только часть входящих дуг, и возникнет конфликт типа «тупик». Условие выполнения вершины слияния вычисляется как дизъюнкция условий активации входящих дуг (рис. 8).

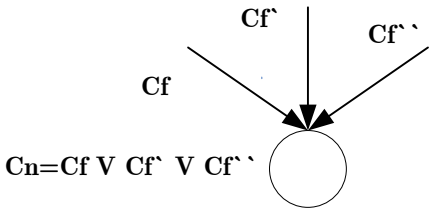

Рисунок 8. Условие активации вершины слияния

Если условия активации входящих дуг попарно взаимоисключают друг друга: $Cf \,\&\, Cf` = false$, $Cf` \&\, Cf`` = false$, $Cf \,\&\, Cf` = false$, то при выполнении потока работ может быть активировано не более одной входящей дуги вершины слияния, иначе возникнет конфликт типа «недостаток синхронизации».

В общем случае операции построения и сравнения булевых функций, которые выполняются в ходе алгоритма верификации, имеют экспоненциальную вычислительную сложность, так как длина вектора значений булевой функции экспоненциально зависит от количества переменных (количества вершин выбора). Тем не менее, основываясь на методе упрощения условий выполнения операторов для последовательных программ [20], условия выполнения вершин и активации дуг можно представить таким образом, что вычислительная сложность алгоритма верификации будет полиномиальна - они должны быть представлены как дизъюнкция идентификаторов вершин выбора или их отрицаний. Рассмотрим рисунок 9, на котором приведен пример работы алгоритма верификации Графа анализа. В вершине-задании $L_t = T2$ возник конфликт типа «тупик», так как условия активации входящих дуг: *true* и $!C2$ не совпадают. Конфликт типа «недостаток синхронизации» определяется в вершине слияния $M2$, так как условия активации входящих дуг: $C2$ и $!C1$ не являются взаимоисключающими. В вершине слияния $M1$ нет «недостатка синхронизации», при этом условие активации этой вершины может автоматически представляется в сокращенной форме: $!C2 \vee C2 = !C1$, $!C1$ - условие выполнения вершины выбора с идентификатором $C2$.

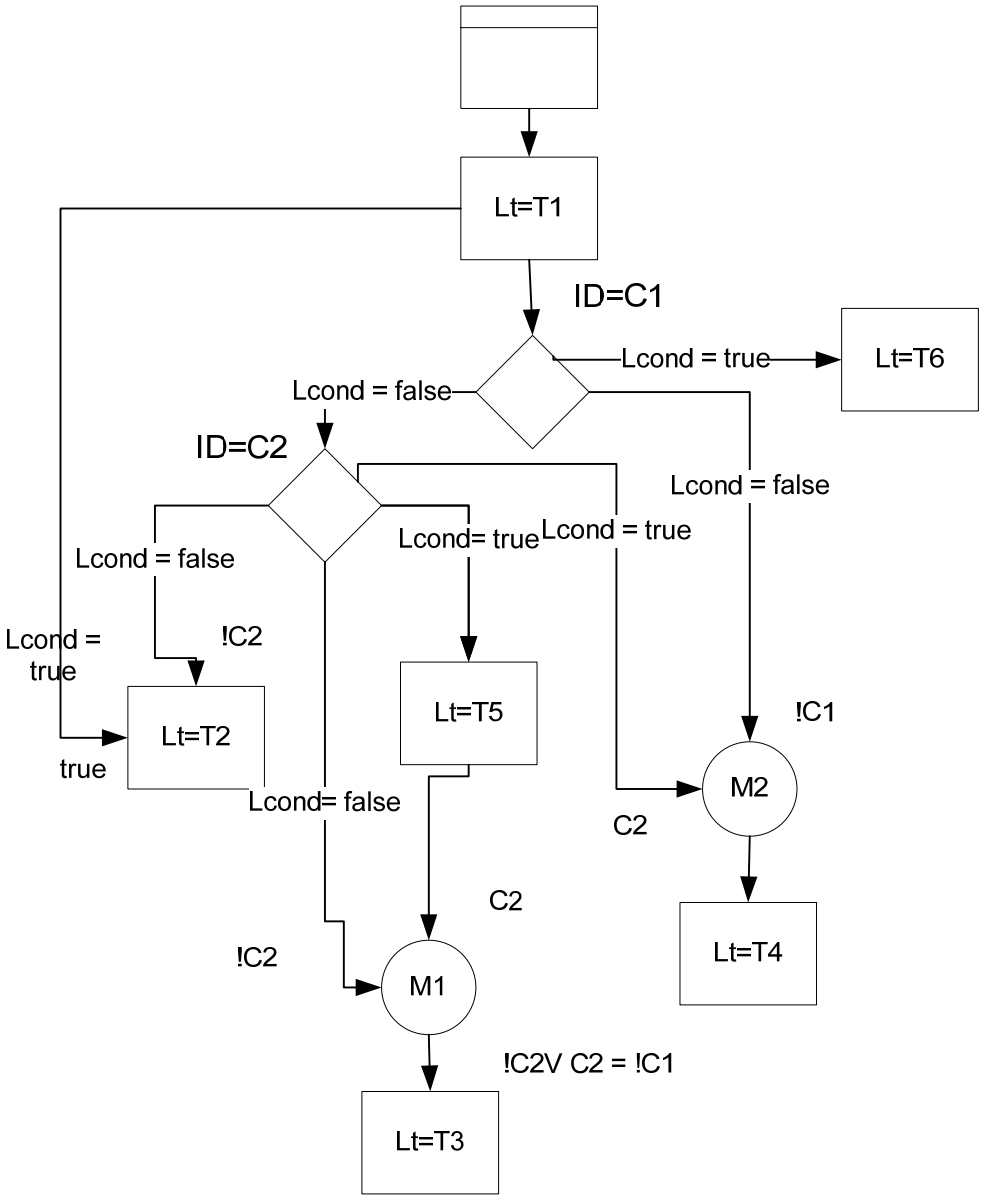

Рисунок 9. Пример работы алгоритма верификации

В ходе разметки графа и проверки его на наличие структурных конфликтов определяются условия выполнения заданий, что дает также возможность предложить алгоритм оптимизации (распараллеливания независящих по данным действий) для потока работ, не содержащего структурных конфликтов. Алгоритм оптимизации в Системе для тела каждого цикла работает следующим образом:

1) Определяются зависимости по данным для заданий и вершин выбора.
2) Проводится поиск в ширину, удаляются избыточные потоки управления между вершинами. Если у вершины удалены все входящие потоки управления, она соединяется новыми входящими потоками управления с вершинами выбора, началом цикла или начальным заданием так, что условие ее выполнения сохраняется.
3) Восстанавливаются потерянные зависимости по данным таким образом, что условия выполнения вершин сохраняются, и избыточные потоки управления не создаются.
4) Если у вершины выбора были потеряны исходящие потоки управления с пометками $L_{cond} = true$ или $L_{cond} = false$, то вершина выбора соединяется с новым

пустым заданием (синхронизатором) исходящим потоком управления с этой пометкой.

5) Вершины, находящиеся внутри цикла и не имеющие исходящих потоков управления, соединяются с вершиной, завершающей тело цикла.

Пример использования Системы автоматической верификации и оптимизации потоков работ для анализа процесса сканирования и публикации книги

В Системе автоматической верификации и оптимизации потоков работ в виде графа анализа был изображен процесс сканирования и публикации книги в рамках ЭБ «Научное наследие России». В ходе работы процесса верификации при представлении графа в виде иерархии вложенных циклов был обнаружен конфликт «мертвый цикл» (рис. 10). Была добавлена недостающая вершина слияния, проведена повторная верификация и выявлен конфликт «тупик» (рис. 11). Избыточный поток управления, из-за которого возникал конфликт, был удален и получен верный граф анализа потоков работ (рис. 12). На рисунке 13 показан граф анализа потока работ после проведенной оптимизации.

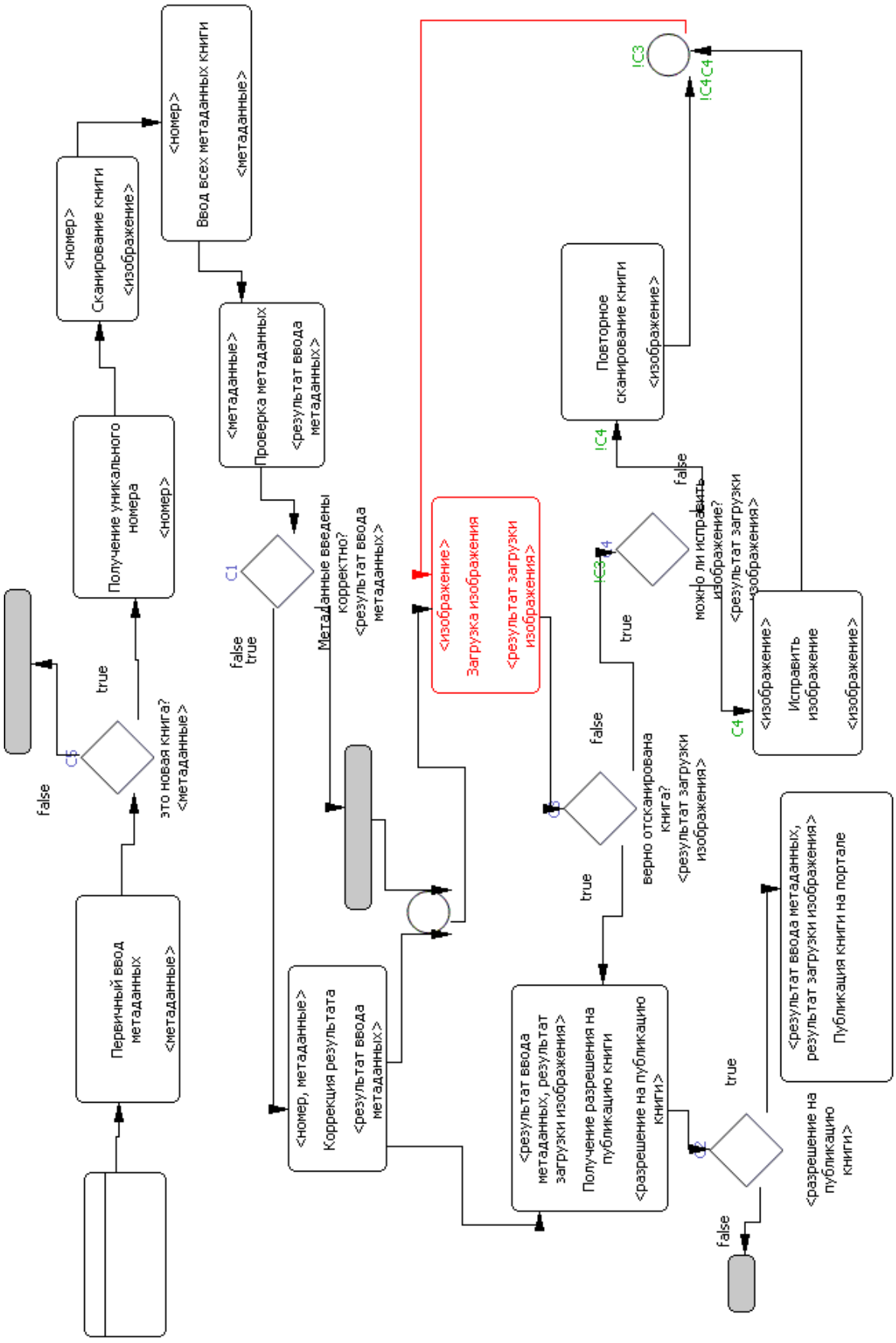

Рисунок 10. Процесс сканирования и публикации книги (конфликт «мертвый цикл»)

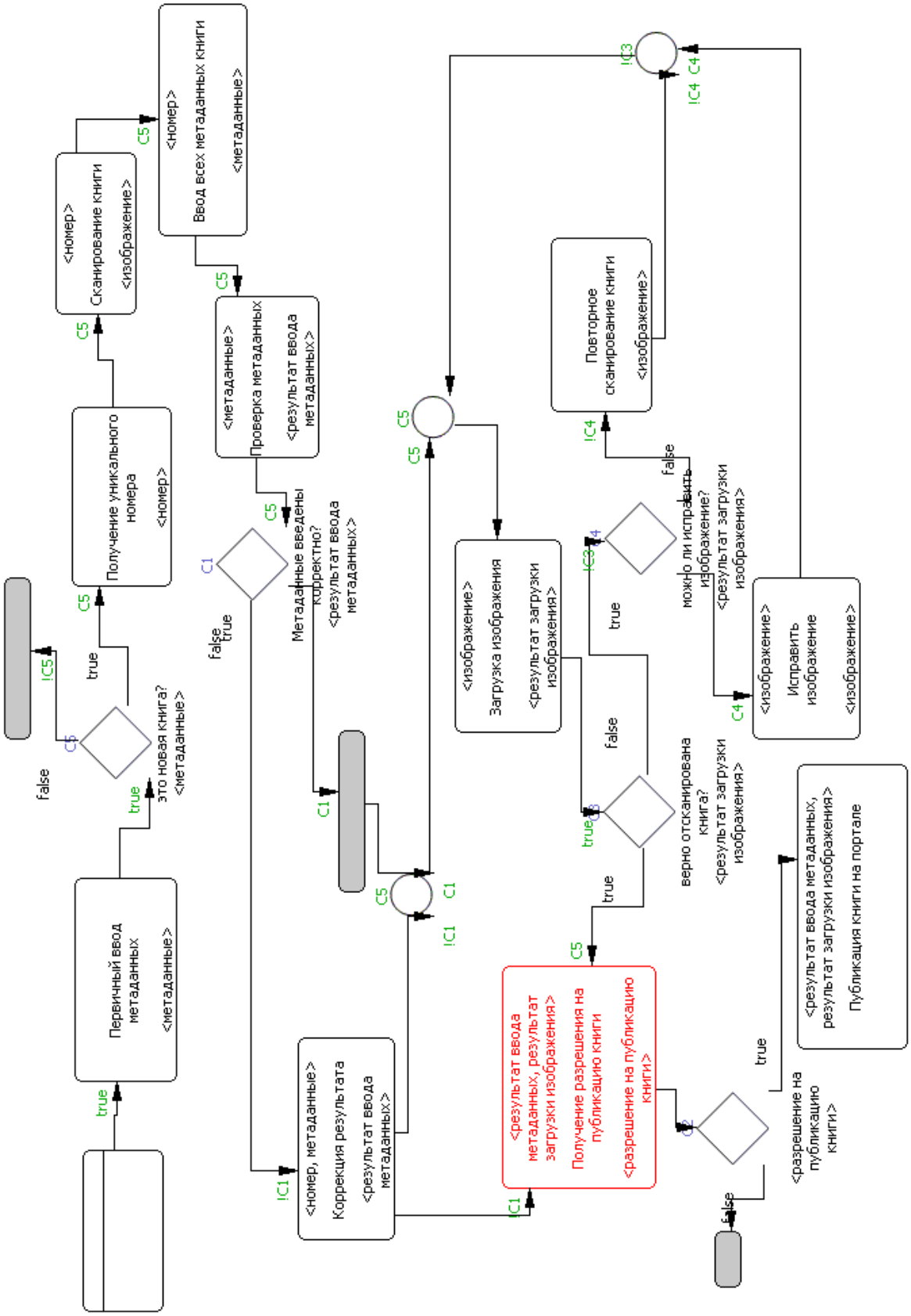

Рисунок 11. Процесс сканирования и публикации книги (конфликт «тупик»)

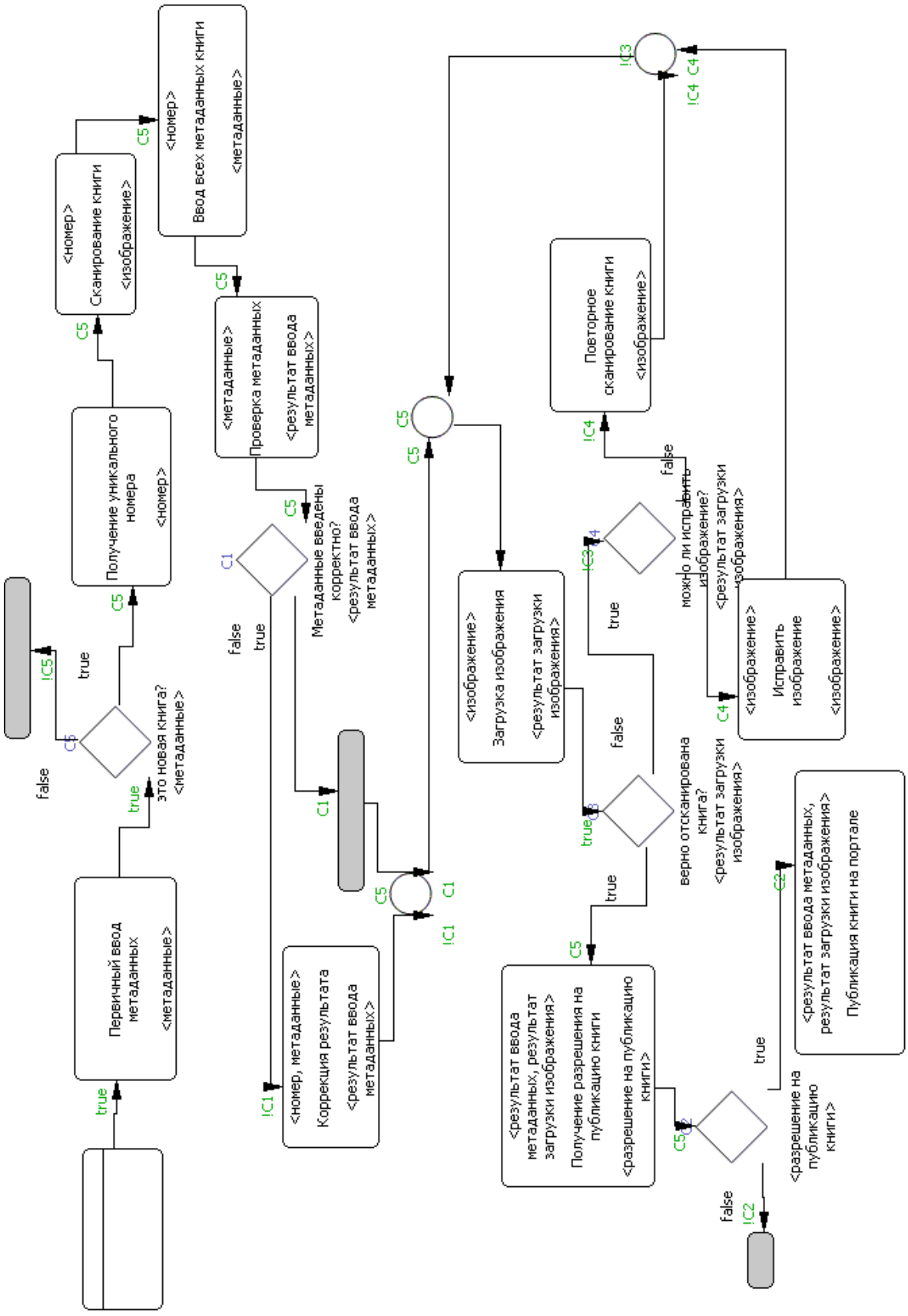

Рисунок 12. Процесс сканирования и публикации книги (граф без структурных конфликтов)

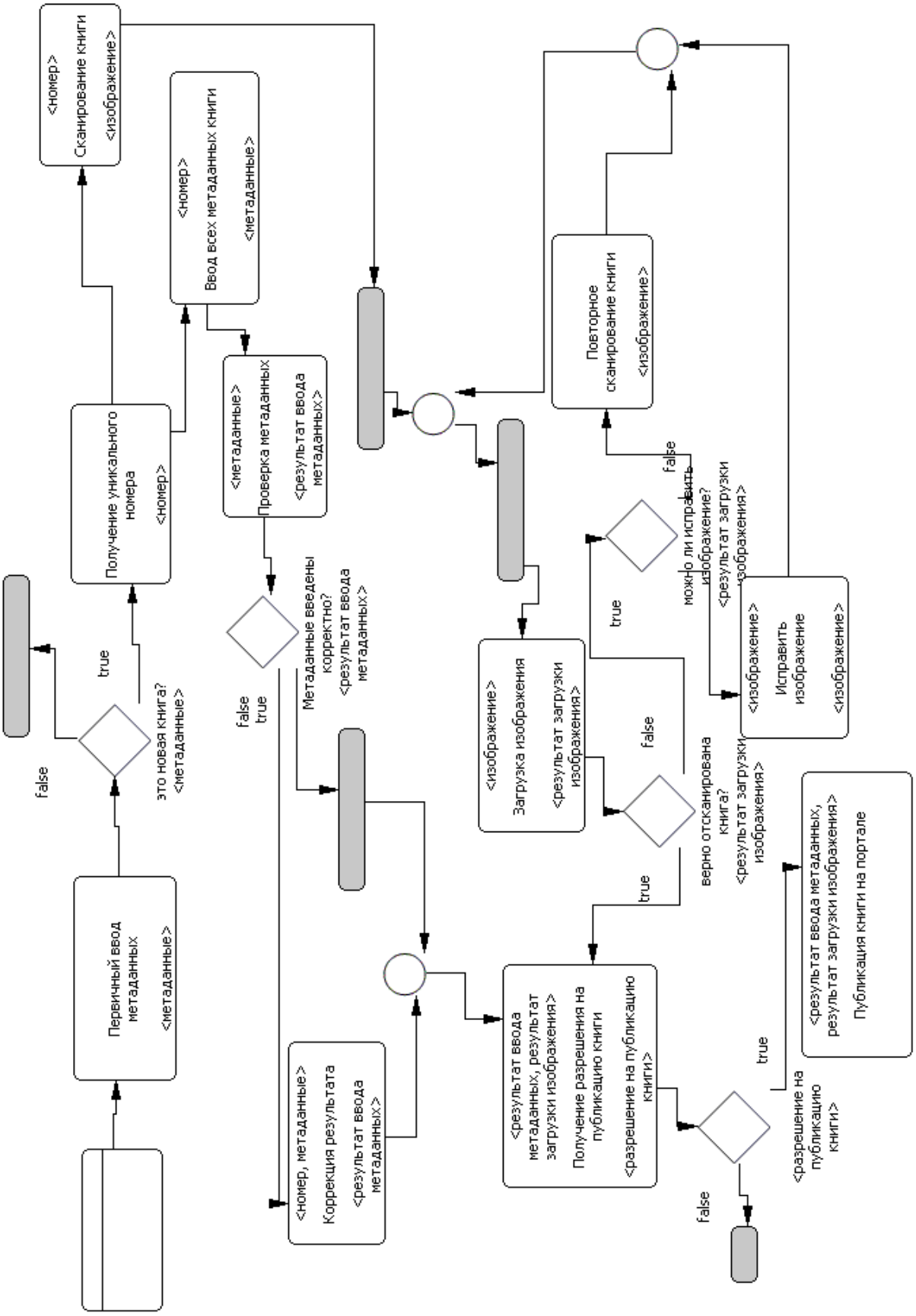

Рисунок 13. Процесс сканирования и публикации книги (оптимизированный поток работ)

Заключение

Нами были разработаны новые методы автоматической верификации и оптимизации потоков работ [10], имеющие полиномиальную вычислительную сложность. Особенность предлагаемых методов заключается в том, что они позволяют работать с иерархией вложенных циклов, на каждом уровне вложенности верифицировать и оптимизировать произвольные перекрывающиеся структуры, которые содержат условные переходы. Была создана Система автоматической верификации и оптимизации потоков работ [11], реализующая эти методы. Теперь нашей целью стал анализ процессов работы с информационными ресурсами в рамках проекта ЭБ «Научное наследие России» [12]. Были рассмотрены рабочие процессы в электронных библиотеках, которые имеют сложною и запутанную структуру, и была обоснована целесообразность использования Системы автоматической верификации и оптимизации потоков работ для их анализа. В дальнейшие планы входит полная автоматизация процессов верификации и оптимизации потоков работ, заданных с помощью языка BPEL4People [16].

Литература

1. Terminology & Glossary / Workflow Management Coalition. – 1999. – URL: http://www.wfmc.org/Download-document/WFMC-TC-1011-Ver-3-Terminology-and-Glossary-English.html.
2. WSO2 . – 2010. – URL: http://wso2.org/.
3. Tempo Intalio. – 2009. – URL: http://www.intalio.org/confluence/display/TEMPO/Home.
4. *Aalst W.M.P.*, *Hirnschall A*., *Verbeek H.M.W*. An Alternative Way to Analyze Workflow Graphs // Electronic Commerce Research – 2002. – V.2, №.3 – P. 195–231.
5. *Lin H., Zhao Z., Li H., Chen Z.* A Novel Graph Reduction Algorithm to Identify Structural Conflicts // Proceedings of the 35th Annual Hawaii International Conference on System Sciences – 2002. – V.9 – P. 289.
6. *Bi H.H., Zhao J.L.* Applying Propositional Logic to Workflow Verification // Information Technology and Management. – 2004. – V. 5, №. 3-4. – P. 293-318.
7. *Толстов Е.В.* Задачи моделирования потоков работ при помощи сетей Петри. – Диссертация на соискание степени кандидата технических наук: 05.13.18 / Е.В. Толстов. – Москва, 2006. – 145 С.
8. *Netjes M.*, *Reijers H.A.*, *Aalst W.M.P.* On the Formal Generation of Process Redesigns // First International Workshop on Model-Driven Engineering For Business Process Management – 2008. – P. 49–60.
9. *Cao H., Jin H., Wu S., Tao Y*. PGWFT: A Petri Net Based Grid Workflow Verification and Optimization Toolkit // Advances in Grid and Pervasive Computing. Third International Conference – 2008. – P. 48–58.
10. *Kalenkova A.A*. Application of If-Conversion to Verification and Optimization of Workflows // Programming and Computer Software. – 2010. – V.36, №. 5 – P.276-288.
11. *Каленкова А.А., Серебряков В.А., Бездушный А.Н.* Система автоматической верификации и оптимизации потоков работ // Телематика: Труды XVII всероссийской научно-методической конференции. – Санкт-Петербург, 2010. – С. 381-382.
12. *Захаров А.А., Филиппов В.И.* Поддержка цифровых библиотек и музейных объектов в среде ЕНИП // Электронные библиотеки: перспективные методы и технологии, электронные коллекции: Труды XI Всероссийской научной конференции RCDL’2009. – Петрозаводск, 2009. – С.487.

13. Business Process Model and Notation (BPMN). FTF Beta 1 for Version 2.0 / Object Management Group – 2009. – URL: http://www.omg.org/cgi-bin/doc?dtc/09-08-14.pdf.

14. Process Definition Interface - XML Process Definition Language 0.03 / Ed. by Marin M., Norin R., Shapiro R. – 2001. – URL: http://xml.coverpages.org/XPDL20010522.pdf.

15. Web Services Business Process Execution Language Version 2.0 / Ed. by Alves A., Arkin A., Askary S. – 2007. – URL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.

16. WS-BPEL Extension for People specification, v1.0 / Ed. by Agrawal A., Amend M., Das M. – 2007. – URL: http://public.dhe.ibm.com/software/dw/specs/ws-bpel4people/BPEL4People_v1.pdf.

17. *Aalst W.M.P., Hofstede A.H.M., Kiepuszewski B., Barros A.P.* Workflow Patterns // Distributed and Parallel Databases – 2003. – V.14, №.3 – P. 5–51.

18. *Каленкова А.А.* Оптимизация потоков работ по времени выполнения, основанная на удалении избыточных потоков управления // Труды МФТИ – 2009. – Т.1, №. 2 – С.160–175.

19. *Евстигнеев В.А., Касьянов В.Н.* Сводимые графы и граф-модели в программировании. – Новосибирск.: ИДМИ, 1999.

20. *Kennedy K., Allen J.R.* Optimizing compilers for modern architectures: a dependence-based approach. – San Francisco: Morgan Kaufmann Publishers Inc., 2001