МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М. В. ЛОМОНОСОВА

На правах рукописи

Терёхина Ирина Юрьевна

Методы выявления аномалий в условиях смеси технологических процессов, сопровождающих наблюдаемый объект

Специальность 2.3.6 —
«Методы и системы защиты информации, информационная безопасность»

Диссертация на соискание учёной степени
кандидата физико-математических наук

Научный руководитель:
доктор физ.- мат. наук, проф.
Грушо Александр Александрович

Москва — 2024

Оглавление

Стр.

Стр.

Введение

Актуальность темы

Диссертация посвящена исследованию подходов к решению задачи поиска отклонений от некоторого принятого режима функционирования (далее аномалий) реальных объектов и процессов их сопровождения. Под процессом в контексте настоящей работы понимается описание информационной технологии, как множество конечных последовательностей атомарных инструкций, предназначенное для сопровождения объектов и/или услуг для пользователя (создание, контроль состояния и управление).

В диссертации рассматриваются объекты, процессы сопровождения которых имеют техническую природу и могут быть формально описаны. Также предполагается, что рассматриваемые объекты могут быть описаны в рамках математических моделей, используемых в контексте диссертации, и вопросы обеспечения их безопасности от деструктивных воздействий, а, значит, вопрос поиска аномалий имеет практическое значение.

В концептуальном плане модели поиска аномалий, представленные в диссертации, разрабатываются в контексте решения двух задач. Первая из них направлена на построение математической модели процесса по доступным логам (стереотипным режимам) функционирования процесса. Вторая обеспечивает поиск аномалий с использованием построенной математической модели процесса, которая описывает легальное (принятое, традиционное) его поведение.

Поиск аномалий процесса с использованием восстановленной по его логам модели является актуальным вопросом с позиции требований обеспечения информационной безопасности процесса по перечисленным далее причинам.

1. Необходимость обеспечения информационной безопасности возникает в большом количестве прикладных областей — от многопользовательских систем, где задача поиска аномалий может возникать в рамках обеспечения целостности данных и разграничения доступа к ним; до областей, где происходит сбор и

систематизация большого количества отчетов (логов), с последующим поиском подходов к выявлению и обнаружению вторжений.

2. Если есть алгоритм построения модели процесса по его логу, и полученная модель отражает возможные варианты исполнения процесса, то этот факт означает, что найден прототип этого процесса без дополнительной информации о том, как соответствующая процессу информационная технология устроена “внутри”. Таким образом, информационная технология может стать более интерпретируемой и понятной для восприятия человеком.
3. В информационной безопасности достаточно сложно дать формальное определение того, что именно можно назвать “аномалией”. Автор диссертации исходит из следующего интуитивного предположения: если известна модель “правильного” функционирования системы или процесса, то определение аномального поведения для системы или процесса может быть дано следующим образом — это такое поведение, которое не может быть порождено этой моделью. В этом предположении дано, как и определение того, что можно назвать аномалией, так и понятен подход к построению алгоритма, осуществляющего поиск аномалий. Такое определение аномалий означает, что при известной модели процесса, задача поиска аномалий может быть переформулирована в задачу соответствия некоторого исполнения процесса построенной модели, решение которой, как правило, является менее трудоемким.

Вопрос построения математической модели по данным некоторой функционирующей информационной технологии (процесса, рабочего процесса) часто формулируется как задача построения модели (обнаружения) процесса. В настоящее время опубликовано и продолжает появляться большое число исследований, посвященных методам восстановления технологических процессов, реализующих информационные технологии, по логам и другим наблюдаемым последовательностям действий, сопровождающих выполнение информационных технологий [1—10]. Впервые подходы к решению этой задачи в контексте технологических процессов представлены в [11], после чего появлялось большое число других подходов к ее решению [12—15]. Традиционно вводятся ряд понятий, которые будут

использоваться и в настоящей диссертации. Понятие *одного исполнения процесса* — наблюдение за изменением атрибутов процесса в ограниченных временных рамках. Исполнению процесса ставится в соответствие понятие *трассы* — непустого слова в некотором конечном непустом алфавите возможных *действий* процесса, наблюдений за изменением конечного подмножества атрибутов, упорядоченных по времени. Восстановленная модель процесса напрямую зависит от заданного *лога* (лога исполнений процесса, лога событий) — конечного неупорядоченного множества трасс, которое предполагается достаточным для описания функционирования процесса.

Для одного процесса, сопровождающего функционирование некоторого объекта, могут быть построены различные математические модели, описывающие этот процесс. Также выбор подходящей математической модели может зависеть от того, поиск решений каких именно задач относительно данного процесса необходимо осуществить. Для задач обеспечения информационной безопасности можно привести следующие примеры того, что может задавать модель процесса.

– Модификация ресурсов субъектами, влияющими на реализацию информационной технологии. В этом случае исследование подходов к решению задачи поиска аномалий сводится к поиску неправомерного использования ресурсов, то есть к построению методов решения задачи управления/разграничения доступа к информационным ресурсам.
– Модификация данных во времени по мере реализации информационной технологии. Тогда вопрос поиска аномалий может быть сведен к вопросу обеспечения целостности данных.

Цели и задачи

Целью диссертационной работы является разработка моделей и алгоритмов для решения задачи поиска аномалий в условиях функционирования нескольких процессов.

Для достижения поставленной цели необходимо решение следующих задач.

1. Поиск эффективных методов моделирования процесса по некоторому логу и методов выявления аномалий, с использованием известной модели процесса.

2. Разработка эффективного по времени выполнения относительно проверяемой на аномальность трассы метода обнаружения аномалий с помощью различных типов математических моделей процесса.
3. Разработка метода построения моделей нескольких процессов в зависимости от свойств лога, описывающего исполнения процессов.
4. Получение оценок сложности алгоритмов решения задачи построения моделей процесса и алгоритмов обнаружения аномалий с помощью моделей процесса, проведение экспериментов.
5. Обобщение алгоритмов решения задачи поиска аномалий на случай, когда проверяемая трасса содержит в себе данные нескольких процессов.

Основные положения, выносимые на защиту

На защиту выносятся обоснование актуальности решаемой задачи, методология, принятая для исследования, научная новизна, теоретическая и практическая значимости работы, а также следующие положения, которые подтверждаются результатами исследования, представленными в Заключении диссертации.

1. Результаты исследования возможности использования математических моделей в виде сетей Петри для решения задачи построения модели процесса и для решения задачи поиска аномалий.
2. Результаты исследования возможности использования математических моделей в виде ациклических ориентированных графов (DAG) для решения задачи построения модели процесса и для решения задачи поиска аномалий.
3. Алгоритмы и соответствующие им оценки сложности выполнения по времени для решения задачи построения моделей нескольких процессов в виде ациклических ориентированных графов.
4. Алгоритмы и соответствующие им оценки сложности выполнения для решения задачи выявления аномалий в некоторой трассе при использовании моделей процессов в виде ациклических ориентированных графов. Обобщение предложенных алгоритмов и оценка сложности выполнения для случая, когда в проверяемой на аномальность трассе содержатся данные нескольких процессов.

Научная новизна

Предложено решение задачи поиска аномалий с помощью математических моделей, описывающих представления нескольких одновременно функционирующих технологических процессов.

Предложены решения и получены оценки временной сложности восстановления множества экземпляров информационных технологий в форме ациклических ориентированных графов. Используя построенные модели процесса в виде ациклических ориентированных графов, предложено решение для задачи поиска аномалий. Предложено обобщение методов восстановления моделей процессов в случаях функционирования нескольких информационных технологий.

Получены результаты, демонстрирующие отсутствие возможности однозначного восстановления модели процесса, представленной в терминах простейших сетей Петри. Этот факт исключает возможность применения данного математического аппарата для решения задачи поиска аномалий.

Разработан и описан подход, позволяющий выбирать математические модели для решения конкретных задач информационной безопасности на примере задачи поиска аномалий. Показано, что критерий простоты описания математической модели не может быть решающим в выборе подходящей модели.

Все результаты, полученные в диссертации, являются новыми.

Теоретическая и практическая значимость

Основная теоретическая значимость исследования состоит в том, что автор предлагает новые решения задач распознавания и построения процессов, описывающих функционирование информационных технологий, используя аппарат глубокого анализа эквивалентных представлений этих процессов.

Практическая значимость полученных результатов состоит в возможности применения результатов в анализе защищенности реальных систем.

Методология и методы исследования

В работе используются методы дискретной математики, теории графов, теории сложности управляющих систем.

Соответствие паспорту научной специальности

Полученные в диссертации результаты соответствуют паспорту специальности 2.3.6 — методы и системы защиты информации, информационная безопасность (физико-математические науки).

Диссертация представляет результаты исследований в области информационной безопасности. В работе используются методы дискретной математики, теории графов, теории сложности управляющих систем.

Области исследования:

3. Методы, модели и средства выявления, идентификации и классификации угроз нарушения информационной безопасности объектов различного вида и класса.
15. Принципы и решения (технические, математические, организационные и др.) по созданию новых и совершенствованию существующих средств защиты информации и обеспечения информационной безопасности.
16. Модели, методы и средства обеспечения внутреннего аудита и мониторинга состояния объекта, находящегося под воздействием угроз нарушения его информационной безопасности.

Достоверность

Достоверность полученных результатов обеспечивается приведенными точными математическими доказательствами. Работы других авторов, используемые в диссертации, отмечены соответствующими ссылками. Результаты диссертации опубликованы в открытой печати.

Апробация работы

Результаты диссертации докладывались на следующих конференциях и научно-исследовательских семинарах.

1. Семинар «Компьютерная безопасность» под руководством д.ф.-м.н., проф. Грушо А.А., д.т.н., проф. Тимониной Е.Е, кафедра информационной безопасности факультета вычислительной математики и кибернетики МГУ им. М.В. Ломоносова, неоднократно с 2017 по 2021 год.
2. Семинар «Компьютерная безопасность» под руководством к.ф.-м.н. Галатенко А.В., к.ф.-м.н. Александрова Д.Е., кафедра МаТИС механико-математического факультета МГУ им. М.В. Ломоносова, 2022 год.

3. Кафедральный семинар кафедры информационной безопасности под руководством д.т.н., акад. РАН Соколова И.А., факультет вычислительной математики и кибернетики МГУ им. М.В. Ломоносова, март 2022 года.
4. Четырнадцатый международный семинар «Дискретная математика и ее приложения» имени академика О. Б. Лупанова, Москва, МГУ им. М.В. Ломоносова, 20–24 июня 2022 год.
5. Семинар «Проблемы современных информационно-вычислительных систем» под руководством д.ф.-м.н., проф. В.А. Васенина, механико-математический факультет МГУ им. М.В. Ломоносова, 7 марта 2023 год.
6. Научная конференция «Ломоносовские чтения 2023», Москва, МГУ им. М.В. Ломоносова. Секция «Вычислительной математики и кибернетики», 4-14 апреля 2023 год.

Личный вклад

Автор сформулировал и доказал представленные в диссертационной работе утверждения, теоремы, направленные на решение задачи поиска аномалий и решение задачи построения формальных моделей процессов. Все результаты получены автором самостоятельно.

Публикации

Результаты, выносимые на защиту, изложены в 4 статьях ([68—70; 73]), 3 из которых опубликованы в рецензируемых научных изданиях, рекомендованных для защиты в диссертационном совете МГУ имени М. В. Ломоносова по специальности 2.3.6, а также в 2 сборниках трудов конференций ([71; 72]).

Объем и структура работы

Диссертация состоит из введения, трёх глав, заключения. Полный объём диссертации составляет 122 страницы, включая 43 рисунка и 2 таблицы. Список литературы содержит 73 наименований.

Краткое содержание работы

В **Введении** обосновывается актуальность исследований, проводимых в рамках диссертационной работы, формулируются цели и задачи работы, излагается научная новизна.

Первая глава посвящена обзору существующих результатов для решения задачи построения модели процесса и задачи поиска аномалий.

Вторая глава посвящена использованию математических моделей в виде сетей Петри для решения задачи поиска аномалий. Основные результаты главы представлены в публикациях [68; 69].

В этой главе проведен анализ требований, предъявляемых к сетям Петри, для решения задачи поиска аномалий, включая:

- свойство надежности;
- свойство, позволяющее обеспечить запрет на последовательное выполнение в сети конструкций синхронизации и выбора;
- полноту лога рабочего процесса для рассматриваемой сети;
- сохранение в сети отношения каузальности между переходами, если между соответствующими действиями в логе это отношение было выполнено.

Приведен пример простого процесса, состоящего из 4 действий, $L = \{ACD, AACD, ABCD\}$. В процессе есть повтор действия A и конструкция “ИЛИ”, допускающая переход без выполнения какой-либо действия (трасса AC). Для этого процесса рассмотрены 5 моделей, $\mathbb{N}_1$, $\mathbb{N}_2$, $\mathbb{N}_3$, $\mathbb{N}_4$, $\mathbb{N}_5$, представленных в виде сети Петри. Доказан ряд утверждений относительно упомянутых выше свойств, которые задают базовые требования корректных сетей Петри для дальнейшего решения задачи поиска аномалий. Показано, почему каждую из моделей нельзя назвать подходящей под эти базовые требования. В таблице Теоремы 1 приведены результаты доказательств утверждений о свойствах корректности для рассмотренных моделей.

Теорема 1. *Результаты главы 2 могут быть представлены с помощью следующей таблицы:*

Сеть Петри	*Надежность*	*SWF-сеть*	*Полнота лога*	*Т. о кауз.*
$\mathbb{N}_1$	+	—	—	+
$\mathbb{N}_3$	+	—	+	—
$\mathbb{N}_4$	—	+	—	—
$\mathbb{N}_5$	+	+	+	—

Показано, что условия Теоремы о каузальности [16] не выполняются, когда в сети Петри есть переходы с одинаковыми действиями, однако, возможно добиться выполнения других свойств корректности с помощью перехода к более широкому классу сетей Петри (нетождественного определения отображения между переходами сети Петри и действиями лога τ). На примере сети $\mathbb{N}_2$ показано, что Теорема о каузальности не выполняется и при наличии блоков *OR-split*, *OR-join*, *AND-split*, *AND-join*. Это говорит о том, что решение задачи поиска аномалий с использованием модели в виде сети Петри, возможно только для простых процессов без ветвлений (“ИЛИ”) и требований одновременного выполнения нескольких условий (“И”).

Таблица в Теореме 1 с точки зрения задачи поиска аномалий свидетельствует о том, что для некоторого процесса (на примере простого процесса, состоящего из 4 возможных действий) наложение нескольких необходимых для дальнейшего поиска аномалий условий корректности приводит к задаче поиска подходящей модели. При этом решение этой задачи является переборным [17] и не гарантирует успешного результата. Тем самым строго сформулировать задачу поиска аномалий, используя для этого модель в терминах сети Петри, не всегда удается.

Для эффективного решения задачи поиска аномалий необходимым условием является наличие модели, которая бы описывала поведение процесса. Более того, необходимо, чтобы данная модель восстанавливалась по полному логу единственным образом и при этом не порождала бы новых трасс, которых не было в исходном логе. Однако приведенный выше пример процесса показывает, что сложности возникают не только с построением единственно возможной модели процесса, но и с произвольной моделью, которая бы удовлетворяла базовым свойствам корректности, которые обычно налагаются на модель в терминах сетей Петри. Таким образом, результатом главы 2 является демонстрация сложности однозначного восстановления

модели в терминах сетей Петри с наложением нескольких условий корректности. Этот факт не позволяет в дальнейшем решить задачу обнаружения аномалий.

Третья глава посвящена восстановлению экземпляров информационных технологий, используя аппарат ориентированных ациклических графов. Построены обобщения методов восстановления моделей процессов в случаях реализации или одновременного функционирования нескольких информационных технологий. Основные результаты главы представлены в публикациях [70; 73].

В работе [11] были предложены три алгоритма и рассчитана временная сложность построения модели в виде ациклического ориентированного графа, в зависимости от различных свойств лога исполнений одного процесса. Пусть $m = |L|$ — количество трасс в логе, $n = |V|$ — количество возможных действий процесса P. Предполагается, что $m \to \infty$, $n \to \infty$. Основным предположением о входных параметрах алгоритмов является предположение, что количество трасс m в логе L превышает количество действий n, $m >> n$. Под элементарными операциями для вычисления оценок сложности алгоритмов понимаются простейшие операции редактирования над графами (например: добавление/удаление вершины, добавление/удаление ребра, слияние/расщепление вершин). Данные результаты были переформулированы для удобства их использования в задаче поиска аномалий и используются в настоящей главе в виде лемм:

Лемма 1 (О сложности построения конформного DAG специального вида)**.** *Пусть задан лог процесса L. Известно, что L содержит только трассы, в которых каждое действие алфавита V встречается ровно по одному разу. Известно, что конформный логу L граф G не содержит циклов. Сложность алгоритма построения конформного графа G составляет $O(mn^2)$.*

Лемма 2 (О сложности построения конформного DAG)**.** *Пусть задан лог процесса L. Известно, что конформный логу L граф G не содержит циклов. Сложность алгоритма построения конформного графа G составляет $O\left(mn^3\right)$.*

Лемма 3 (О сложности построения конформного графа)**.** *Пусть задан лог процесса L. Сложность алгоритма построения конформного логу L графа G составляет $O\left(m\left(kn\right)^3\right)$, где k – максимальное количество повторов некоторого действия в логе.*

Основными этапами предложенных алгоритмов, которые строят модель процесса в виде графа по заданному логу L, являются:

1. добавление в граф всех дуг, соответствующих зависимостям между действиями, представленными в логе L;
2. удаление дуг внутри сильно связных компонент графа;
3. добавление меток на дуги, которые присутствуют в транзитивных замыканиях подграфов, порождаемых трассами лога L. Удаление дуг без меток;
4. склейка вершин, соответствующих одинаковым действиям.

Первая задача, результаты решения которой представлены в этой главе, является построение формальных моделей нескольких процессов, по заданному логу возможных исполнений этих процессов. Сначала рассматривается случай, когда известно, что одна трасса лога может содержать данные ровно одного процесса. Далее рассмотрен случай, когда в рамках одной трассы лога могут встречаться данные нескольких процессов.

Вторая задача, решение которой описано в этой главе, направлена на поиск аномалий в трассе с использованием построенных ранее моделей процессов. Рассматривается случай, когда проверяемая на аномальность трасса является исполнением только одного процесса, так и когда проверяемая на аномальность трасса содержит в себе исполнение нескольких процессов.

Рассматривается s процессов $P_1,\ldots,P_s$, с непустыми множествами действий $V_1,\ldots,V_s$ и графами $G_1 = (V_1,E_1),\ldots,G_s = (V_s,E_s)$ соответственно.

Пусть задан некоторый лог L, $M = |L| \to \infty$, $n = \max_i |V_i| \to \infty$, $i = 1,\ldots,s$, k – максимальное количество повторов действия в логе L. Также, как и в работе [11], предполагается, что $M >> n$, количество процессов s процессов конечно.

Теорема 2 (Процессы с различными множествами действий)**.** *Пусть задан лог L для s процессов. Пусть выполнены условия Лемм 1, 2, 3. Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \varnothing$, при $i \neq j, i, j = 1, \ldots, s$. Тогда сложность построения s конформных графов составляет $O\left(Mn^2\right), O\left(Mn^3\right), O\left(M\left(kn\right)^3\right)$ соответственно, в зависимости от свойств лога L.*

Если построено s конформных графов зависимостей для s процессов и известно, что в поступающей трассе ω может быть исполнение только одного процесса, то задача поиска аномалий в ω сводится к задаче определения согласованности этой трассы с s графами зависимостей.

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$,$i = 1, \ldots, s$, $u = |\omega|$ — длина трассы ω, $u \to \infty$.

Теорема 3 (Поиск аномалий в процессах с различными множествами действий)**.** *Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \varnothing$, при $i \neq j, i, j = 1, \ldots, s$. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за не более, чем $s + C_1 un + C_2 en^2$ операций, где C_1, C_2 — некоторые константы.*

Если для задачи поиска аномалий, построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 1 (Поиск аномалий в процессах с различными множествами действий)**.** *Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \varnothing$, при $i \neq j, i, j = 1, \ldots, s$. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за $O\left(u\right)$.*

Пусть $n = \max_i |V_i| \to \infty, i = 1, \ldots, s$. Пусть $m = \max_i |L_i| \to \infty$, L_i – часть лога L, соответствующая i-му процессу, $i = 1, \ldots, s$, k — максимальное количество повторов действия в логе L. Если известно, что процесс однозначно можно идентифицировать его начальным действием, то при условии s различных начальных действий, справедлива следующая Теорема:

Теорема 4 (Отличимые процессы по начальному действию)**.** *Пусть задан лог L для s процессов. Пусть s процессов имеют попарно различные начальные действия. Тогда сложность построения s конформных графов составляет $O\left(mn^2\right)$, $O\left(mn^3\right)$, $O\left(m\left(kn\right)^3\right)$ соответственно, в зависимости от свойств лога L.*

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$,$i = 1, \ldots, s$, $u = |\omega|$ — длина трассы ω, $u \to \infty$.

Теорема 5 (Поиск аномалий в процессах, отличимых по начальному действию)**.** *Пусть s процессов имеют попарно различные начальные действия. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за не более, чем $s + C_1 un + C_2 en^2$ операций, где C_1, C_2 — некоторые константы.*

Если для задачи поиска аномалий, построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 2 (Поиск аномалий в процессах, отличимых по начальному действию)**.** *Пусть s процессов имеют попарно различные начальные действия. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за $O\left(u\right)$.*

Рассмотрен случай, когда не налагается условие на попарно различные начальные действия для каждого из s процессов, но есть возможность разделить лог L на части по одному уникальному для процесса действию. Таким образом, обеспечивается необходимое и достаточное условие существования системы различных представителей для теоремы Ф. Холла. Пусть $n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$. Пусть $m = \max_i |L_i| \to \infty$, L_i – часть лога L, соответствующая i-му процессу, $i = 1, \ldots, s$, k — максимальное количество повторов действия в логе L.

Теорема 6 (Отличимые процессы по некоторому действию)**.** *Пусть задан лог L для s процессов. Пусть каждый из s процессов имеет по одному*

известному действию A_i, $i = 1, \ldots, s$ *такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей* i*-му процессу. Тогда сложность построения* s *конформных графов составляет* $O\left(mn^2\right)$, $O\left(mn^3\right)$, $O\left(m\left(kn\right)^3\right)$ *соответственно, в зависимости от свойств лога* L.

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$,$i = 1, \ldots,s$, $u = |\omega| \to \infty$ — длина трассы ω.

Теорема 7 (Поиск аномалий в процессах, отличимых по некоторому действию)**.** *Пусть каждый из* s *процессов имеет по одному известному действию* A_i, $i = 1, \ldots, s$ *такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей* i*-му процессу. Пусть для* s *процессов построено* s *конформных графов зависимостей, тогда поиск аномалий в некоторой трассе* ω *может быть выполнен за не более, чем* $s + C_1 un + C_2 en^2$ *операций, где* C_1, C_2 *— некоторые константы.*

Если для задачи поиска аномалий, построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 3 (Поиск аномалий в процессах, отличимых по некоторому действию)**.** *Пусть каждый из* s *процессов имеет по одному известному действию* A_i, $i = 1, \ldots, s$ *такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей* i*-му процессу. Пусть для* s *процессов построено* s *конформных графов зависимостей, тогда поиск аномалий в некоторой трассе* ω *может быть выполнен за* $O\left(u\right)$.

Теорема 4 рассматривает случай уникальных для процессов подстрок длины 1, которые начинаются с первой буквы трасс лога L. Пусть $m = \max_i |L_i| \to \infty, n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$, k — максимальное количество повторов действия в логе L. Справедливо обобщение этой теоремы:

Теорема 8 (Отличимые процессы по конечной подстроке)**.** *Пусть задан лог* L *для* s *процессов. Пусть известно, что каждая из трасс лога,*

начиная с некоторого номера i содержит подстроки длины r, каждая из которых может принадлежать только одному из s процессов. Тогда сложность построения s конформных графов составляет $O\left(mn^2\right)$, $O\left(mn^3\right)$, $O\left(m\left(kn\right)^3\right)$ соответственно, в зависимости от свойств лога L.

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$,$i = 1,\ldots,s$, $u = |\omega| \to \infty$ — длина трассы ω.

Теорема 9 (Поиск аномалий в процессах, отличимых по конечной подстроке)**.** *Пусть известно, что трасса ω, начиная с некоторого номера i содержит подстроку длины r, которая может принадлежать только одному из s процессов. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в трассе ω может быть выполнен за не более, чем $C_1 un + C_2 en^2$ операций, где C_1, C_2 — некоторые константы.*

Если для задачи поиска аномалий, построены модели процессов в виде ациклических ориентированных графов, тогда параметры n, e будут константами и справедливо:

Следствие 4 (Поиск аномалий в процессах, отличимых по конечной подстроке)**.** *Пусть известно, что трасса ω, начиная с некоторого номера i содержит подстроку длины r, которая может принадлежать только одному из s процессов. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в трассе ω может быть выполнен за $O\left(u\right)$.*

Пусть теперь в логе L в одной трассе могут содержаться данные нескольких процессов.

Пусть $M = |L| \to \infty$, $n = \max_i |V_i| \to \infty$, $i = 1,\ldots,s$, $M >> n$. Пусть множества действий для процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1,\ldots,s$.

Теорема 10 (Множество процессов в трассе без повторов действий)**.** *Пусть задан лог L для s процессов. В каждой из трасс лога L могут*

присутствовать действия от 1 до s процессов и в рамках одной трассы нет повторов действий. Пусть конформные графы для s процессов не содержат циклов. Пусть множества действий для процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$. Если для всех действий $A \in V_i, B \in V_j, i \neq j$, i, $j = 1, \ldots,$ s в логе L выполнено: $A \lesssim B$ и $B \lesssim A$, то сложность построения s конформных графов составляет $O\left(Mn^2\right)$.

Если построенные s конформных графов по Теореме 10 обладают свойствами из Теорем 2, 4, 6, 8 и некоторая трасса $\boldsymbol{\omega}$ содержит в себе исполнение только одного процесса, то для ответа на вопрос, является ли трасса $\boldsymbol{\omega}$ аномальной, могут быть применимы Следствия 1, 2, 3, 4 соответственно.

Следующая теорема отвечает на вопрос о сложности поиска аномалий, когда и поступающая трасса $\boldsymbol{\omega}$ содержит в себе исполнения нескольких процессов:

Теорема 11 (Поиск аномалий для трассы, содержащей исполнения множества процессов, имеющих различные множества действий)**.** *Пусть множества действий для s процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$. Пусть для s процессов построено s конформных графов зависимостей. Пусть в некоторой трассе $\boldsymbol{\omega}$ могут содержаться исполнения множества процессов. Тогда поиск аномалий в трассе $\boldsymbol{\omega}$ может быть выполнен за не более, чем $s(s + C_1 un + C_2 en^2)$ операций, где C_1, C_2 – некоторые константы.*

Если для задачи поиска аномалий построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 5 (Поиск аномалий для трассы, содержащей исполнения множества процессов, имеющих различные множества действий)**.** *Пусть множества действий для s процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$. Пусть для s процессов построено s конформных графов зависимостей. Пусть в некоторой трассе $\boldsymbol{\omega}$ могут содержаться исполнения множества процессов. Тогда поиск аномалий в трассе $\boldsymbol{\omega}$ может быть выполнен за $O\left(u\right)$.*

Можно рассчитать нижнюю границу количества действий в логе L по всем трассам для ограничений, налагаемых в Теореме 10. Пусть $R = \sum_{\omega \in L} |\omega|$ — общее количество действий в логе, $n_i = |V_i|, i = 1, \ldots, s$ — количество действий каждого из s процессов.

Следствие 6. *Пусть задан лог L, который не содержит повторов действий. Пусть в каждой из трасс лога L могут присутствовать действия от 1 до s процессов. Пусть для всех действий $A \in V_i, B \in V_j, i \neq j, i, j = 1, \ldots, s$ в логе L выполнено: $A \lesssim B$ и $B \lesssim A$. Тогда $|L| \geqslant 2$, $R \geqslant 2 \sum_{i=1}^{s} n_i$.*

В этой главе рассмотрена задача построения формальной модели множества процессов в виде ациклического ориентированного графа и задача поиска аномалий с помощью построенных эталонных моделей процесса. Найдены ограничения на возможности применения алгоритмов для одного процесса и определены оценки сложности построения формальных моделей для множества процессов в зависимости от свойств лога.

Для задачи построения моделей процесса рассмотрены случаи наличия исполнений нескольких процессов как между трассами одного лога, Теоремы 2, 4, 6, 8, так и внутри трасс одного лога, Теорема 10.

Показано, что с помощью системы различных представителей возможно эффективно выделять траектории множества различных процессов, Теорема 6.

Для задачи поиска аномалий предложены алгоритмы ее решения. Показано, что оценка сложности предложенных алгоритмов является линейной относительно длины трассы, для которой требуется дать ответ об аномальности, и квадратичной относительно максимально возможного количества действий и зависимостей в одном моделируемом процессе, Теоремы 3, 5, 7, 9, 11.

В **Заключении** приведены основные результаты работы, которые заключаются в следующем.

1. Исследована возможность использования математических моделей в виде ациклических ориентированных графов (DAG) для решения задачи построения модели процесса и для решения задачи поиска аномалий.

Показано, что для моделей, сформулированных в терминах ациклических ориентированных графов, удается успешно решать задачу поиска аномалий при условии наличия нескольких одновременно функционирующих процессов. В том числе продемонстрировано, что с помощью системы различных представителей можно эффективно выделять траектории множества различных процессов.

2. Получены временные оценки сложности построения моделей процесса при описании модели процесса с помощью ациклических ориентированных графов.

   При этом продемонстрировано, что в ряде случаев алгоритмы восстановления нескольких одновременно функционирующих процессов полностью повторяют их аналоги для одного функционирующего процесса, что позволяет использовать уже готовые оценки сложности для решения задачи поиска аномалий.
3. Получены временные оценки сложности выявления аномалий при описании модели процесса с помощью ациклических ориентированных графов, которые являются линейными относительно длины проверяемой трассы.
4. Показано, что использование формального аппарата в виде сетей Петри для решения задачи поиска аномалий затруднено тем фактом, что модель процесса не всегда удается однозначно восстановить.

   Налагая простые требования корректности, такие, как: свойство надежности; свойство, позволяющее обеспечить запрет на последовательное выполнение в сети конструкций синхронизации и выбора; полнота лога рабочего процесса для рассматриваемой сети; сохранение в сети отношения каузальности между переходами, если между соответствующими действиями в логе это отношение было выполнено — не гарантирован результат построения хоть какой-либо подходящей однозначной, корректной модели процесса.

   Продемонстрировано, что без перехода к более сложному классу сетей Петри класс технологических процессов, для которого

возможно решить задачу поиска аномалий с использованием восстановленной моделью процесса, сильно ограничен.

5. Показано, что не каждая математическая модель описания реального процесса является подходящей для эффективного решения задач информационной безопасности. Тем самым продемонстрирован подход, позволяющий выбирать модели для конкретных задач информационной безопасности.

Благодарности

Автор выражает благодарность и большую признательность своему научному руководителю доктору физико-математических наук, профессору Грушо Александру Александровичу за постановку задач, а также сотрудникам кафедры информационной безопасности факультета вычислительной математики и кибернетики МГУ имени М. В. Ломоносова за внимание к работе.

Автор благодарит семью, друзей и коллег за оказанную поддержку в процессе написания работы.

Глава 1
Обзор существующих результатов для задачи обнаружения процесса и задачи поиска аномалий

В настоящей диссертации под *технологическим процессом* имеется в виду описание функционирования некоторой информационной технологии, множество конечных последовательностей атомарных инструкций, предназначенное для создания материального или информационного продукта, или предоставления услуги для пользователя. Под *обнаружением или построением модели процесса* понимается метод определения некоторого структурированного описания процесса из множества примеров. Под подходящим формальным языком для структурированного описания процесса понимается язык, который является математически однозначно определенным и такой, что модель на этом языке обеспечивает строгое определение динамического поведения процесса.

В контексте задачи обеспечения информационной безопасности, которая в работе является определяющей, задача построения модели процесса рассматривается совместно с задачей обнаружения аномалий. При этом задача поиска аномалий может формулироваться как для поиска недостатков самой модели, описывающей процесс, так и для определения того, являются ли аномальными новые поступающие последовательности инструкций, *исполнения процесса*, которые происходят в режиме реального времени. В работе основное внимание сосредоточено на решении задачи поиска аномалий для новых исполнений процесса в предположении, что построенная формальная модель описывает все легальные возможные исполнения процесса “достаточно хорошо”.

В этой главе описываются и сравнительно анализируются существующие подходы для решения задачи построения модели процесса и задачи поиска аномалий.

1.1 Задача построения модели процесса

Задача построения модели процесса не является новой, однако все еще остается актуальной. В контексте технологических процессов решение этой задачи впервые представлено в работе [11], после чего появлялось большое число подходов к ее решению [12—16].

Традиционно, для задачи построения модели процесса вводятся понятие *одного исполнения процесса* — наблюдение за изменением атрибутов процесса в ограниченных временных рамках. Исполнению процесса ставится в соответствие понятие *трассы* в некотором конечном алфавите возможных *действий* (инструкций) процесса, как наблюдение за изменением конечного подмножества атрибутов, упорядоченных по времени. После этого появляется возможность ввести определение понятия *лога* (лога событий) — конечного неупорядоченного множества трасс, достаточных для адекватного описания функционирования процесса и дальнейшего восстановления математически строгого описания модели процесса.

Существует бесконечное множество моделей процесса, которые могли бы порождать выделенное в лог множество трасс, поэтому у задачи построения модели процесса существует, в общем случае, более одного решения. Каждый алгоритм построения модели процесса ищет некоторое решение в заранее обозначенном множестве возможных моделей процесса. Заранее обозначенное множество возможных моделей процесса обычно подразумевает построение моделей с использованием одного из возможных формальных моделей и математического аппарата, на котором они написаны. Существуют решения для задачи построения модели процесса, где в качестве формальной модели рассматриваются конечный автомат [18; 19], сеть Петри [16; 20], вероятностные модели [12; 21; 22], ориентированный граф [11; 23; 24], множество ассоциативных правил [13; 25] и другие.

В работе [12] обсуждается проблема разработки формальных моделей для сложных программных процессов и впервые формулируется задача построения модели процесса, как метод анализа данных, которую авторы называют *process discovery*. Задача решается посредством грамматического

вывода с использованием ДКА. Но также рассмотрен метод решения, основанный на цепях Маркова и возможность применения нейронных сетей для решения задачи построения процесса. В работе подчеркивается, что задача построения процессов является актуальной в контексте автоматизации создания формальных моделей для облегчения внедрения технологических процессов.

Существует большое количество таксономий для алгоритмов построения модели процесса. Одна из них — деление по количеству трасс, порождаемых моделью, к их числу относятся: *точное* или *эвристическое (шумное)* построение модели процесса, в зависимости от того, способна ли модель породить те же самые трассы, из которых она была построена изначально.

Так, алгоритм эвристического построения модели процесса, предложенный, например, в работе [12], не порождает трассы, включение которых в модель может сделать ее слишком сложной или слишком общей. В этом случае алгоритм посчитает эти трассы *шумом*. В таком случае становится актуальным решение задачи: что делать с такими “шумными” трассами – отбросить их не всегда является лучшей идеей, они должны хотя бы частично поддерживаться моделью. А модель, в свою очередь, должна уметь порождать и шумные трассы, где некоторые действия в шумных трассах могут быть, в каком-то смысле неправильными с точки зрения модели. Таким образом, необходимо определение метрики, которая задавала бы отношение между трассой и построенной моделью. Данная метрика в работах обычно встречается, как *соответствие* или *конформность* трассы модели. Конформность трассы модели показывает, какая часть трассы может быть порождена моделью. Решение задачи построения модели в эвристическом подходе может сводиться, например, к построению такой модели, в которой трассы лога обладают наибольшей суммарной конформностью модели. Использование алгоритмов эвристического построения модели процесса для решения задачи аномалий в рамках диссертации рассматриваться не будет.

Объяснение того факта, почему та или иная трасса лога была порождена моделью, также является актуальной областью исследований, особенно для случаев, когда модель представляет собой модель машинно-

го обучения (области “explainable AI” или “interpretable machine learning”), [26; 27].

Существуют работы, посвященные сравнительному анализу подходов к моделированию процессов: в [28] рассматривается задача выбора класса формальных моделей в контексте трудоемкости использования той или иной модели с течением времени; в [29] анализируется возможность применения языка UML для построения моделей процесса; в [30] исследуется вопрос моделирования со стороны понимания языка человеком, было обнаружено, что язык EPC (Event-driven Process Chains [31]) выглядит для человека более понятным, чем язык сетей Петри. При этом существуют работы [32], которые показывают, что глубокое знание конечным пользователем языка, на котором представлена модель, не является обязательным условием.

Упорядочивание моделей процесса по отношению “эта модель имеет больше смысла для описания процесса, чем другая модель” в литературе встречается, как *задача уместности модели* [33]. В рамках решения этой задачи было отмечено, что модели могут слишком детально отражать специфику процесса или, наоборот, быть слишком общими. Для описания этих случаев вводятся термины *переобучение* или *недообучение*. Так, например, если речь идет о модели в терминах машинного обучения, то, традиционно, определение понятия переобучения используется, когда модель слишком “заточена” на те данные, на которых она обучалась, и, поэтому имеет более низкую точность классификации на новых данных. Определение понятия недообучения для моделей машинного обучения используется, когда существует много трасс, порождаемых моделью, но которые не присутствовали в логе, на котором модель была обучена [26].

Не существует общего соглашения насчет того, что считать приемлемым балансом между специфичностью и общностью модели. Обычно, определяется отношение частичного порядка $\succ$ относительно не только моделей, но и лога, с помощью которого эти модели были построены, $\langle M_1, L\rangle \succ \langle M_2, L\rangle$ [33]. В таком случае, алгоритм построения модели ищет модель, которая является максимумом относительно введенного отношения порядка среди моделей.

В работе [18] рассматривается решение задачи построения модели процесса, где модель строится в виде конечных детерминированных автоматов с минимальным количеством состояний. В работе показано, что задача построения модели процесса, как на только позитивных (трасса, которая соответствует модели процесса), так и одновременно на позитивных и негативных (трасса, которая точно не соответствует модели процесса) примерах, является NP-трудной.

В работе [34] была предложена вероятностная модель и определена область "вероятно близких решений" (PAC-learnable domain) задачи построения модели процесса, в эту область вошли классы булевых функций и деревьев решений. Показано, что грамматики, включая детерминированные конечные автоматы, не входят в область PAC-learnable.

В работе [35] рассматривается задача восстановления регулярного множества из примеров, предложен полиномиальный алгоритм для конечных детерминированных автоматов с оракулом. При этом возможными вопросами для оракула являются: "Принимается ли данная строка правильной грамматикой?", "Эта грамматика является правильной?".

В работе [36] впервые предлагается решение задачи в виде построения префиксного дерева.

В работе [37] в рамках анализа ДНК/РНК последовательностей были предложены эффективные алгоритмы для выявления паттернов и построения модели процесса. Построенная модель при этом принимает класс регулярных выражений, в которых содержатся операторы выбора и итерации. Также авторы расширили работу алгоритма на корректную обработку шумных данных с помощью вычисления метрики похожести строк. В качестве метрики рассматривается расстояние Левенштейна и метрика, использующая матрицы аминокислотных замен.

В области машинного обучения одними из первых работ являются работы [21; 22], которые предлагают решение задачи построения модели процесса в виде рекуррентных нейронных сетей, с последующим анализом скрытого слоя для поиска состояний и переходов полученной грамматики.

Помимо теоретических результатов, существует большое количество программных продуктов, которые включают в себя решение задачи построения модели процесса, например [38; 39].

Вместе с задачей построения модели процесса исследуется *задача переобнаружения процесса* [16]: возможно ли построить другие модели процесса, используя уже построенные модели этого процесса? Будут ли новые модели совпадать с уже построенными моделями? Иллюстрация показана на Рис. 1.

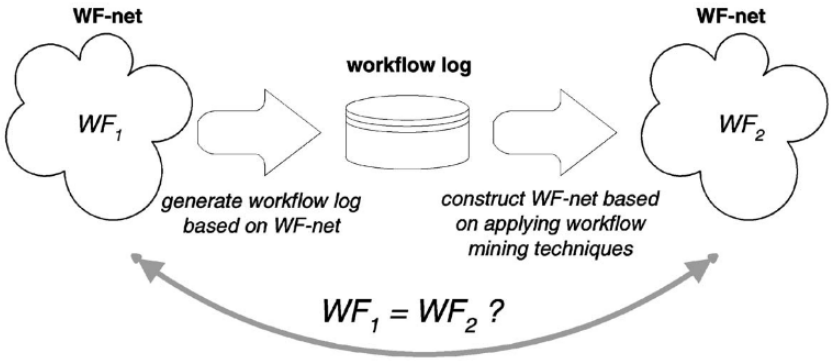

Рис. 1 — Задача переобнаружения процесса [16].

Пусть есть лог, порожденный моделью рабочего процесса WF_1. Применяя к этому логу некоторый алгоритм майнинга, строится новая модель WF_2. Задача состоит в том, чтобы выделить класс моделей таких, что $WF_1 = WF_2$. Равенство моделей при этом рассматривается как равенство сетей Петри с точностью до переименования мест. Задача переобнаружения процесса используется для определения теоретических границ и тестирования алгоритмов построения модели процесса. Было показано [16], что не существует алгоритма переобнаружения процесса, который был бы способен переобнаружить все возможные модели в терминах сетей Петри, удовлетворяющих заранее обозначенным условиям корректности.

Для постановки задачи построения модели процесса, как правило, исходят из следующих интуитивных предположений:

- возможно собрать логи технологических процессов с данными о событиях;
- каждое событие относится к некоторой задаче (шаг рабочего процесса);
- каждое событие относится к примеру (исполнению) рабочего процесса;
- события упорядочены по времени (при том, что задачи могут исполняться параллельно).

Основной трудностью в построении точной модели процесса является получение конечного лога, по которому эта модель строится. Если рассматривать процесс, содержащий большое число альтернативных и параллельных исполнений, то нет гарантий, что произвольный лог содержит все возможные комбинации исполнений этого процесса. Например, при параллельном выполнении 10 задач число возможных перестановок составляет $10! = 3628800$ и, таким образом, маловероятно, что каждая из перестановок присутствует в логе. Более того, некоторые исполнения процесса могут иметь меньшую вероятность возникновения, чем другие, тем самым, оставаясь не обнаруженными. Также трудностей в решение задачи построения модели добавляет наличие шума в данных (редкие события, исключения/ошибки, некорректно записанные данные). Перечисленные факты являются примерами проблемных вопросов, возникающих при построении формальной модели, адекватно описывающей процесс.

С другой стороны, если построенной модели присущ параллелизм или циклы, то количество трасс, которые она может порождать, может быть неограниченным. Даже если модель не содержит циклов и параллелизма, но в ней есть N условных переходов, то количество трасс для формирования полного лога, по которому эта модель может быть воспроизведена, имеет экспоненциальный порядок.

Таким образом, идеальным для задачи построения процесса является случай, когда есть возможность наблюдать все исполнения процесса, тем самым, порождая полный лог, строить по этому логу модель, и этой моделью, в свою очередь, иметь возможность порождать все возможные трассы, которые могут быть в процессе.

Из открытых подзадач для более эффективного решения задачи построения модели процесса также можно отметить задачи: обеспечения отсутствия зацикливания действий процесса; обеспечения отсутствия действий, которые никогда не выполнятся; наличия мусора после завершения выполнения всех действий процесса и другие.

В работе [40] анализируется связь между структурой модели бизнес-процесса и степенью ее интерпретируемости/понимания. Авторы отмечают следующие вопросы, требующие осмысления и разрешения.

– Необходимость построения не произвольных моделей, а моделей, которые были бы хорошо интерпретируемы и могли бы поддерживаться специалистами без высокого уровня компетенции [41].
– Существующие решения моделирования бизнес-процессов либо строят слишком абстрактные модели (например, фреймворк SEQUAL [42]), либо, наоборот, слишком сложные модели с большим количеством зависимостей.
– Существует недостаток теоретических исследований, посвященных не только построению, но и поддержке уже построенной модели бизнес-процесса.
– Существуют работы, посвященные оценке качества построенной модели ([43], [44], [45], [46]), но практически нет работ, посвященных тому, как на практике необходимо строить качественную модель.

Как результат, предложены правила построения моделей процесса.

1. Использовать минимально возможное количество элементов.
2. Минимизировать количество путей, проходящее через один элемент.
3. Использовать одно стартовое и одно завершающее события.
4. Модель должна быть структурирована насколько это возможно. Модель процесса называется структурированной, если для каждого “разбивающего” логического соединения существует соответствующее ему “объединяющее” логическое соединение (например, блоку AND-split с 3 выходящими ребрами соответствует блок AND-join с 3 входящими ребрами).
5. Избегать блоки с неисключающим “ИЛИ”. Модели, имеющие только логические соединения “И” и “XOR”, подвержены ошибкам меньше [47]. Кроме того, существуют недостатки в семантике блоков OR-join, что ведет к парадоксам и трудностям имплементации [48].
6. Использование стиля “глагол-существительное” в наименовании меток.
7. Декомпозиция модели, если она имеет более 50 элементов. Для моделей, имеющих более 50 элементов, было показано, что вероятность ошибки в них превышает 0.5 [47].

Эти правила были получены, в основном, в результате эмпирических наблюдений, анализа и систематизации. Именно эти предположения использовались автором настоящей диссертации для выбора условий корректности, одновременное выполнение которых рассматривалось для построения модели процесса в виде сетей Петри в главе 2.

1.2 Задача поиска аномалий

Формальное определение того, что можно называть аномалией для процесса, напрямую зависит от того, будет ли строится модель процесса, задающая типичное или атипичное поведение процесса. Если модель процесса будет использоваться, то формулировка задачи поиска аномалий, в свою очередь, зависит от того, с помощью какого математического аппарата задана модель процесса. Если говорить о возможных математических аппаратах, с помошью которых может быть задана модель процесса, то к настоящему времени предложено большое количество подходов к решению задачи построения модели процесса по некоторому логу. Классификацию таких подходов удобнее проводить по тому подходу и математическому аппарату, на основе которого строится модель: конечный автомат; сеть Петри; вероятностная модель, например, Марковские поля; модель классического машинного обучения или нейронная сеть. В данной диссертации в качестве формального аппарата будут рассмотрены сети Петри в главе 2 и ациклические ориентированные графы в главе 3.

Наиболее частые причины аномалий в данных:

- ошибки ввода данных в систему (человеческий фактор);
- ошибки измерений (ошибки инструментальных средств);
- ошибки экспериментов;
- ошибки в обработке данных;
- ошибки семплирования;

– естественные ошибки (неучтенный феномен в данных, коллективные аномалии).

Задача поиска аномалий может быть определена по-разному, например, исходя из того, известно ли исследователю “правильное” поведение системы, которое может быть задано одной или несколькими математическими моделями.

Поиск аномалий без использования модели процесса

В данном разделе будут рассмотрены существующие наиболее эффективные алгоритмы решения задачи поиска аномалий, когда неизвестна модель, описывающая структуру процесса. Показаны ограничения алгоритмов, их достоинства и недостатки. В настоящей диссертации такой способ решения задачи поиска аномалий далее рассматриваться не будет.

В случае задачи поиска аномалий без помощи математической модели процесса лог чаще всего рассматривается в виде табличного представления по различным признакам. Аномалии (или выбросы) в этом случае могут разделяться на одномерные и многомерные, в зависимости от размерности признакового пространства [49]. Под признаком в контексте работы понимается некоторое свойство наблюдаемого объекта. Задача нахождения аномалий сводится к задаче определения распределения (одномерного или многомерного) признака для всех наблюдаемых объектов и задаче определения соответствия признака уже отдельного объекта построенному распределению. Если решение второй задачи не дает положительного ответа о том, что некоторый признак рассматриваемого объекта соответствует известному для этого признака распределению, то весь объект считается аномалией или выбросом.

Возможные методы для нахождения аномалий таким образом [50; 51]:

– z-оценка (Extreme Value Analysis);

– вероятностное и статистическое моделирование (параметрические);

– линейные регрессионные модели (PCA, LMS);
– модели, основанные на близости (Proximity Based Models) (не параметрические);
– модели, основанные на теории информации;
– методы для нахождения аномалий в высокоразмерных разреженных данных (High Dimensional Outlier Detection Methods).

Простые статистические оценки

Примером оценки, которая может применяться для решения задачи поиска аномалий, может служить z-оценка. z-оценка применима при предположении нормального распределения признака. После трансформации значений признака к стандартному нормальному распределению (feature scaling), z-оценка каждой точки случайной величины x может быть вычислена следующим образом: $z = \frac{x-\overline{X}}{S_x}$, где $\overline{X}$ — среднее значение, S_x — стандартное отклонение. Значения $\overline{X}$ и S_x при этом могут быть вычислены по выборочным данным, или получены в генеральной совокупности.

При использовании такой оценки определяется порог, при попадании дальше которого, точки считаются аномалиями. Данный метод прост в реализации и хорош в тех случаях, когда поиск аномалий происходит в низкоразмерных признаковых пространствах.

Методы, основанные на кластеризации

Наиболее популярным алгоритмом, основанным на кластеризации, для задачи поиска аномалий, является алгоритм DBSCAN — Density Based Spatial Clustering of Applications with Noise [52]. DBSCAN — алгоритм, основанный на плотности, применяется для решения задачи поиска аномалий в непараметрических распределениях.

Для алгоритма кластеризации определяется несколько видов точек, к числу которых относятся следующие.

– Внутренняя точка: A — внутренняя точка, если в ее окрестности (которая задается радиусом ε) содержатся не менее $MinPts$ точек, где $MinPts$ — параметр.

– Граничная точка: C — граничная точка, если она принадлежит некоторому кластеру и в ее окрестности менее $MinPts$ точек. Однако при этом она *“достижима по плотности”* (определение следует ниже) из других точек кластера.
– Аномалия: N — не принадлежит никакому кластеру и не *“достижима по плотности”*, а также не *“связна по плотности”* (определение ниже), ни с какой другой точкой. Таким образом, такая точка имеет “свой кластер”.

В начальный момент времени каждая внутренняя точка формирует свой кластер. Для того, чтобы определить, что представляет собой кластер, вводятся следующие далее отношения:

– точка Q *достижима по плотности* из точки P, если существует путь из точек $p_1, \ldots, p_n$, $p_1 = P$, $p_n = Q$, где каждая p_{i+1} лежит в окрестности точки p_i, при этом все точки в пути являются внутренними, кроме, возможно, последней точки Q;
– так как отношение достижимости по плотности не является симметричным, вводится симметричное отношение “связности по плотности”: точки P и Q *связаны по плотности*, если существует точка O такая, что обе точки P и Q достижимы по плотности из O.

Поиск ближайших точек к некоторой точке происходит по плотности в “n-мерной сфере” с радиусом ε. Под кластером понимается максимальное множество соединенных по плотности точек в признаковом пространстве.

Кластер должен удовлетворять следующим двум свойствам: все точки в кластере должны быть взаимно связаны по плотности; и, если точка достижима по плотности из некоторой точки кластера, она тоже должна являться частью кластера.

Алгоритм DBSCAN включает следующие шаги его реализации [53] (Рис. 2):

1. поиск точек в ε окрестности каждой еще не посещенной точки и выделение внутренних точек с более чем $minPts$ соседями;
2. поиск связанных по плотности компонент внутренних точек в списке соседей, при этом игнорируются не внутренние точки;

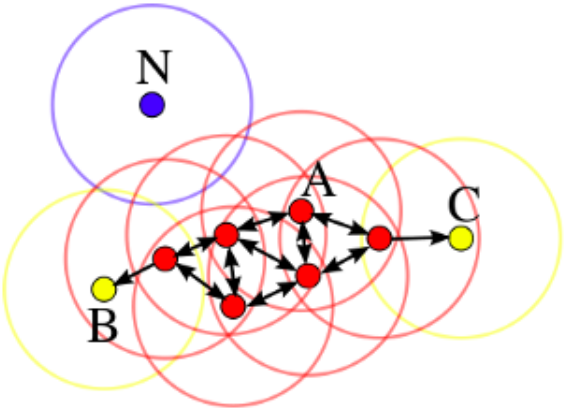

Рис. 2 — Иллюстрация работы алгоритма DBSCAN [52]. Примеры точек при $MinPts = 3$. Внутренние точки обозначены красным, граничные точки — желтым, аномалия — синим.

3. назначение каждой не внутренней точки ближайшему кластеру, если кластер является достижимым по плотности, в противном случае точка считается аномалией.

К преимуществам данного алгоритма можно отнести:

– алгоритм самостоятельно определяет количество кластеров;
– алгоритм подходит для признаковых пространств размерности больше, чем 3;
– возможность выделения аномалий в отдельный кластер, после чего этот кластер можно изучать отдельно/отбросить;
– сложность алгоритма по памяти составляет $O(n)$, где n — общее количество точек;
– сложность работы алгоритма по времени в худшем случае составляет $O(n^2)$, с возможностью улучшения до $O(n \log n)$, где n — общее количество точек, если использовать некоторый индекс для хранения соседей каждой точки и выполнять запрос к соседям точки за $O(\log n)$;
– алгоритм может найти кластеры, которые не являются линейно разделимыми;
– простая визуализация и интерпретация результатов;
– существуют готовые реализации [54] с возможностью задавать различные параметры ε, $MinPts$, метрику близости между точками

(в [54] это по умолчанию евклидова метрика), алгоритм для определения соседей точки.

К недостаткам данного алгоритма можно отнести следующие факторы:

- данные для алгоритма должны быть нормализованы;
- алгоритм сильно чувствителен к выбору параметров ε, $MinPts$ и метрики;
- алгоритм без учителя, поэтому необходимо производить переобучение при поступлении новых данных.

Изолирующие леса

Данный метод основан на бинарных деревьях решений [55]. Основным предположением алгоритма является то, что в данных существует малое количество аномальных точек и все эти точки должны лежать далеко от остальных.

Чтобы построить дерево (этап обучения), алгоритм выбирает произвольно признак из признакового пространства и разбивает его значения по произвольному значению между максимумом и минимумом для этого признака. Данная процедура выполняется для всех объектов обучающей выборки, пока в каждой вершине дерева не останется по одному объекту выборки.

Процедуру можно представить в виде бинарного дерева, где в вершинах будут объекты со значениями одного признака. У левых потомков некоторой вершины значения признака будут меньше, чем значение в вершине, у правых потомков — значения будут больше или равны значению в вершине.

Количество разбиений значений признака, чтобы таким образом *изолировать какую-либо вершину*, равно длине пути от корневой вершины до листа. Этот путь, усредненный по всему лесу аналогичных бинарных деревьев для одного признака — мера нормальности/аномальности этого значения.

Основное предположение для поиска аномалий — случайное разбиение генерирует более короткие пути для аномалий [55]. Если лес из

случайных деревьев по каждому из признаков генерирует короткие пути для некоторого объекта, можно предполагать, что этот объект является аномалией. Метрика аномальности для некоторого объекта x определяется следующим образом:

$$s(x, n) = 2^{-\frac{E(h(x))}{c(n)}},$$

где $h(x)$ — длина пути для объекта x, $E(h(x))$ — среднее $h(x)$ в множестве изолирующих деревьев, $c(n)$ — длина максимального пути в бинарном дереве, n — количество листьев в дереве. Таким образом, вычисление для каждого объекта значения метрики от 0 до 1, где единица констатирует наличие аномальности, 0 — нормальность объекта, позволяет получить решение задачи поиска аномалий.

Достоинствами данного подхода можно назвать следующие:

- для алгоритма не нужна нормализация данных;
- алгоритм подходит для случаев, когда сложно предположить распределение некоторого признака;
- существуют готовые программные реализации [56] с возможностью задавать: количество деревьев в лесу, размер выборки для каждого дерева, предполагаемое количество аномалий от всей выборки (для подбора границы, что считать аномалией), количество признаков для дерева, способ генерации выборки для деревьев (с повторениями или без).

К недостаткам данного подхода можно отнести: трудоемкую визуализацию результатов; а также то обстоятельство, что если удачно не подобраны параметры алгоритма, то это ведет к долгому обучению и большому потреблению памяти.

Поиск аномалий с использованием модели процесса

В данном разделе описаны существующие подходы к решению задачи поиска аномалий, когда известна модель, описывающая процесс.

Определение того, что считать аномалией

Следует отметить сложность в формальном определении того, что можно назвать *аномалией*. Часто авторы (например, [57; 58]) исходят из интуитивных предположений. Например, можно полагать, что “нормальные” данные всегда находятся где-то в “одном месте”, имеют хорошую метрику близости. Такое интуитивное понимание ведет к развитию алгоритмов, основанных на поиске ближайших соседей. С другой стороны, можно предполагать, что аномальные объекты имеют низкую вероятность появления, если построена хорошая генеративная модель “нормальных” объектов. Такое предположение ведет к развитию семейства техник, основанных на статистических моделях.

При определении того, какую из трасс можно считать аномальной, обычно используются следующие содержательные предположения.

- Множество исполнений процесса может быть разделено на множество *нормальных* и *аномальных* исполнений.
- Каждое аномальное исполнение *“нечасто”* во множестве всех исполнений, хотя все множество аномальных исполнений может быть частым.
- Модели процесса, которые *“объясняют”* исполнения из нормального множества, *“имеют смысл”*.
- Модели процесса, которые могут объяснить как нормальные, так и аномальный исполнения, *“имеют меньше смысла”*.

Для того, чтобы использовать такие интуитивные предположения в алгоритмах поиска аномалий, можно привести следующий пример определений терминов “нечасто”, “объясняют” и “имеют смысл”:

- A — конечное непустое множество имен действий, A^* — множество всех слов в алфавите A;
- $t \in A^*$ — трасса;
- $L = \{\langle t, n_t \rangle\}$ — мультимножество, лог действий, где n_t — количество трасс t в логе;
- $size(L) = \sum_{<t,n_t>\in L} n_t$ — размер лога;
- $freq_L(t) = n_t / size(L)$ — частота трассы t в логе L;
- $freq_{\max}$ — константа, задающая понятие "нечасто";

– $M \vdash L$ — задано отношение “объясняет” между моделью процесса M и логом L;
– задан некоторый частичный порядок “имеет больше смысла” между моделями $M_1 \succ M_2$, обозначающий, что M_1 “имеет больше смысла”, чем M_2.

Интуитивные предположения относительно поиска аномалий теперь могут быть псевдо-формализованы следующим образом. Пусть дан лог L:
– L может быть разбит на 2 мультимножества A (аномалии) и N (не аномалии) такие, что $A \cup N = L$ и $A \cap N = \varnothing$;
– $\forall \langle a, n_a \rangle \in A$, $freq_L(a) \leqslant freq_{\max}$;
– пусть M_N — максимальное относительно частичного порядка $\succ$ в $\{M | M \vdash N\}$;
– пусть M_L — максимальное относительно частичного порядка $\succ$ в $\{M | M \vdash L\}$;
– тогда $M_N \succ M_L$.

Оценка качества поиска аномалий

При решении задачи поиска аномалий некоторым алгоритмом необходимо понимать, насколько алгоритм хорош. Откуда вытекает смежная задача — оценка качества работы алгоритма, решающего задачу поиска аномалий. Оценка качества, как правило, происходит с помощью метрик качества.

Чаще всего используемые метрики качества работы алгоритма поиска аномалий вводятся по аналогии с бинарным классификатором [59]. В данном случае “положительный” — решение о трассе было, что трасса аномальная, “отрицательный” — трасса нормальная.

– Истинно-положительный (true positive, TP) — пример (трасса), классифицированный алгоритмом, как положительный, и который в реальности является положительным.
– Ложно-положительный (false positive, FP) — пример, классифицированный, как положительный, но который в реальности является отрицательным.

– Истинно-отрицательный (true negative, TN) — пример, классифицированный, как отрицательный, и который в реальности является отрицательным.
– Ложно-отрицательный (false negative, FN) — пример, классифицированный, как отрицательный, но который в реальности является положительным.

Таким образом, становится возможным рассмотрение стандартных метрик качества бинарного классификатора, которые дают представление о том, насколько алгоритм поиска аномалий хорош/плох.

– $recall/sensitivity/TPR = \frac{TP}{TP+FN}$ — полнота, доля истинно положительных объектов среди всех в реальности положительных объектов.
– $FPR = \frac{FP}{TN+FP}$ — доля ложно положительных объектов среди объектов отрицательного класса, ошибка первого рода.
– $specifity/TNR = \frac{TN}{TN+FP}$ — доля истинно отрицательных объектов среди всех в реальности отрицательных объектов.
– $FNR = \frac{FN}{TP+FN}$ — доля ложно отрицательных объектов среди объектов положительного класса, ошибка второго рода.
– $precision = \frac{TP}{TP+FP}$ — точность, доля истинно положительных объектов среди всех найденных классификатором положительных объектов.
– $accuracy = \frac{TP+TN}{TP+TN+FP+FN}$ — доля правильных ответов алгоритма, бесполезна в задачах с неравными классами.
– $balanced\_accuracy = \frac{TPR+TNR}{2}$ — аналог метрики $accuracy$ для задач с неравными классами.

Алгоритмы решения задачи поиска аномалий

Хорошей иллюстрацией решения задачи поиска аномалий с помощью построения математической модели является работа [58], где описывается 4 алгоритма для нахождения аномалий в логах процессно-ориентированных систем (ПОС). Качество алгоритмов оценивается на множестве из 1500 искусственных логов с различными количеством аномальных трасс и разными их видами присутствия в логе. Было показано, что алгоритм,

основанный на семплировании (описан далее), показывает себя лучше по введенным метрикам качества. Найденные таким образом аномальные трассы могут быть в дальнейшем проанализированы: являются ли они некорректным исполнением некоторого примера, и, если да, далее могут быть проанализированы причины и последствия этого некорректного исполнения. Таким образом, алгоритмы могут быть использованы как первый автоматизированный шаг для более полного аудита безопасности для гибких и нестрогих ПОС.

1. Наивный подход решает задачу определения понятий “объясняет” и “имеет больше смысла” (отношения $\vdash$ и $\succ$) тем, что не использует их совсем. Используя только первые два пункта интуитивных предположений, наивный подход говорит, что трассы аномальны, если они встречаются в логе нечасто. Данный алгоритм помечает нормальные трассы аномальными, если они не встречались часто в логе, однако при этом он не пропустит истинно аномальных трасс. Таким образом, алгоритм дает нулевую ошибку второго рода (false negative rate), но при этом максимальную ошибку первого рода (false positive rate). То есть алгоритм решает задачу поиска аномалий и удовлетворяет требованиям задачи поиска мошеннических действий.
2. Пороговый алгоритм (threshold algorithm) — считает аномальными все трассы, которые имеют соответствие модели ниже заданного порога.
3. Итеративный алгоритм — расширение порогового алгоритма итеративной процедурой. На каждой итерации только трасса с наименьшим соответствием модели помечается, как аномальная, и удаляется из лога на следующей итерации.
4. Семплирующий алогритм — основан на интуитивном предположении, что если аномалии не являются частыми, то случайная выборка трасс из лога их не содержит. Таким образом, модели, полученные из семплированного лога, не включают в себя аномальные трасссы. Трассы из лога, которые не могут быть порождены моделями, полагаются аномальными.

Для алгоритмов из работы [58] необходимо задание некоторого количества параметров. Для всех алгоритмов (кроме наивного) необходимо выбрать алгоритм майнинга процесса. Итеративному алгоритму и пороговому алгоритму необходима метрика для измерения соответствия трассы модели и значение порога. Итеративному алгоритму необходимо задать условие для остановки. Семплирующему алгоритму необходимо задать размер выборки трасс для семплирования. Определение этих параметров чаще всего связано с процессом перебора под конкретную обучающую выборку.

1.3 Поиск аномалий в функционировании технологического процесса

Задача поиска аномалий является одной из классических задач, решение которых актуально в контексте информационной безопасности. Одной из мотиваций ее решения для технологических процессов является необходимость обнаружения мошеннических действий, так как пропуск таких аномалий связан с потерей прибыли/убытками компании. Поэтому на алгоритмы поиска аномалий часто налагается дополнительное условие наличия малой ошибки второго рода (false negative rate). Так, например, если некоторое исполнение процесса несло мошеннический характер и на соответствующей трассе возникла ошибка второго рода, такое исполнение процесса (трасса) будет пропущено и не передано специалистам для дальнейшего выяснения причин и принятия мер, что в свою очередь может вести к убыткам компании. С другой стороны, если предполагать, что анализ человеком каждого примера — дорогая операция, становится необходимым, чтобы алгоритмы имели и малую ошибку первого рода (false positive rate). В таком случае возникает меньше трасс, которые помечены, как аномальные, а на самом деле такими не были. Стоит отметить, что малая ошибка второго рода для этой задачи является более предпочтительной, чем малая ошибка первого рода.

Если же основная мотивация для решения задачи обнаружения аномалий — поиск ошибок в процессах, тогда нет явного предпочтения малой ошибки второго рода, так как это реже связано с потерей прибыли [57]. Поэтому в данном случае должен соблюдаться некоторый баланс между величинами ошибок первого и второго рода.

Если полагать, что аномалии задают мошеннические действия, происходящие в информационной системе, необходимо, чтобы доля ложноположительных объектов была минимальной. Тем самым, необходимо, чтобы метрика $recall$ была близкой к единице. Использовать только эту метрику было бы плохой идеей, так как наивный алгоритм, отсекающий объекты по частоте, имеет максимальное значение $recall$, но при этом имеет минимальное значение $precision$. Поэтому, например, в работе [58] используется метрика, комбинирующая метрики $recall$ и $precision$:

$$f\beta = (1 + \beta^2)\frac{precision \cdot recall}{\beta^2 precision + recall}$$

$f1$-мера — гармоническое среднее между точностью и полнотой. При $\beta > 1$ с ростом β значение полноты влияет на меру больше.

Актуальной открытой задачей также является уменьшение количества ложно-положительных трасс, порождаемых моделью, с целью уменьшения дополнительных расходов на анализ этих трасс человеком. Подходы к решению этой задачи в настоящей диссертации не рассматриваются.

1.4 Выводы

В настоящей главе описаны и проанализированы существующие методы решения задачи построения модели процесса и задачи поиска аномалий. Описаны существующие результаты по этим и смежным задачам, основные предположения, открытые подзадачи.

Описана мотивация выбора условий корректности для модели процесса, которую было бы возможно использовать для поиска решения задачи поиска аномалий, сформулированных в главе 2, и существующие подходы (наивный, пороговый, итеративный, семплирующий) для восстановления моделей процессов в виде ациклических ориентированных графов для главы 3.

Следующие главы посвящены исследованию применимости математических моделей в качестве моделей нескольких одновременно функционирующих процессов в контексте решения задачи поиска аномалий. В качестве формализма для моделей процесса в диссертации рассматриваются сети Петри и ациклические ориентированные графы. Выбор этих классов моделей обусловлен большим количеством публикаций за последние годы, что говорит об актуальности этих подходов для описания моделей процесса. При этом в контексте задач обеспечения информационной безопасности сети Петри не рассматривались, а ациклические ориентированные графы рассматривались только при условии функционирования одного процесса. Ответ же на вопрос применимости математических моделей на примере построения подхода к решению задачи поиска аномалий напрямую переносится на аналогичный вопрос применимости этих моделей и в других задачах обеспечения информационной безопасности, что показывает важность и актуальность результатов, продемонстрированных в главах 2 и 3.

Глава 2
Выявление аномалий с помощью моделей, заданных сетями Петри

Целью главы является исследование возможности использования математической модели в виде выделенного класса сетей Петри для моделирования рабочего процесса. Показано, что при отсутствии ограничения на уникальность действий в логе построить корректную модель рабочего процесса не всегда удается. Показано, что если в модели присутствуют переходы, которые не соответствуют ни одному из действий лога, в частности, вспомогательные переходы $AND\text{-}split$, $AND\text{-}join$, $OR\text{-}split$, $OR\text{-}join$, то нарушаются утверждения, предложенные в работе [16], о взаимосвязи между действиями лога рабочего процесса и переходами соответствующей сети Петри. Тем самым показано, что алгоритм построения модели процесса по логу из работы [16] работает лишь для узкого класса технологических процессов, в которых нет повторов действий и условий “ИЛИ”. Результаты главы представлены в работе [68].

К достоинствам построения моделей на языке сетей Петри [17; 60] можно отнести следующее.

– Сети Петри могут быть использованы для описания параллельно выполняющихся процессов.
– Сети Петри с наложением условия максимального параллелизма являются полными по Тьюрингу [17].
– Возможность моделировать логические операции, если в моделях сетей Петри наложить дополнительное условие ингибиторных (тормозящих) дуг.
– Моделирование в терминах сетей Петри помогает получить дополнительную информацию о структуре и динамическом поведении моделируемой системы.

В работе [16] описываются алгоритмы для построения моделей технологических процессов в терминах сетей Петри по известному логу этого

процесса. Также вводится ряд свойств корректности, налагаемых на сеть Петри для того, чтобы модель могла быть построена по логу функционирования технологического процесса.

В настоящей работе полагается, что в логе нет шумных трасс и лог содержит всю “необходимую” информацию. В качестве моделей используются WF-сети (workflow nets) (формальное определение используемых сетей Петри далее), класс сетей Петри для описания технологических процессов. Рис. 3 — пример такой сети.

2.1 Пример построения модели процесса в виде сети Петри, по его логу

Пусть лог процесса показан в Таблице 1. Таким образом, возможные наблюдения/трассы/исполнения процесса:

```
1 ABCD
2 ACBD
3 ABCD
4 ACBD
5 AED
```

В трассах процесса 1, 2, 3, 4 выполняются 4 действия A, B, C, D. В трассе 5 процесса выполняются 3 действия: A, E, D. Каждая трасса начинается с выполнения действия A, заканчивается выполнением действия D. Если выполнено действие B, то выполнится и действие C. В половине случаев действие C выполняется до действия B. Таким образом, лог этого процесса $L = \{ABCD, ACBD, AED\}$.

Делая некоторые предположения о полноте лога (что все наблюдения репрезентативны и приведено минимальное необходимое подмножество

возможных поведений процесса), возможно сделать модель процесса, показанную на Рис. 3. Модель представлена в терминах сетей Петри [61] (формальное определение дано далее).

Наблюдение	Действие
1	A
2	A
3	A
3	B
1	B
1	C
2	C
4	A
2	B
2	D
5	A
4	C
1	D
3	C
3	D
4	B
5	E
5	D
4	D

Таблица 1 — Лог процесса [16]

Функционирование сети Петри начинается с выполнения действия A и заканчивается выполнением действия D. После выполнения A происходит выбор между либо исполнением параллельно B и C, либо исполнением E. Для выполнения параллельно B и C добавлено 2 новых искусственных действия ($AND\text{-}split$ и $AND\text{-}join$). Эти действия были добавлены с целью маршрутизации и не присутствуют в логе рабочего процесса. Действия называются *параллельными*, если они могут появляться в любом порядке.

Может рассматриваться различная интерпретация мест, переходов и меток в сети Петри. Так, в лекции [60] для моделирования параллель-

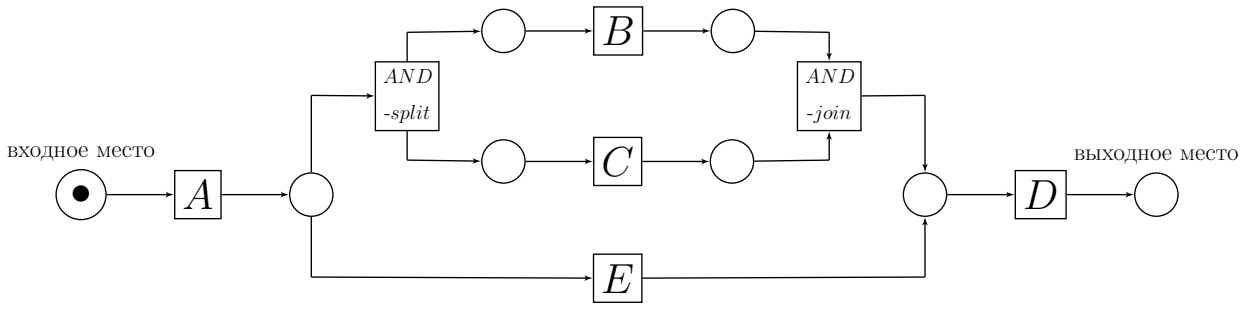

Рис. 3 — Модель процесса, соответствующая логу рабочего процесса из Таблицы 1 [16]

ных процессов предлагается следующая интерпретация узлов сетей Петри: места — регистры памяти, переходы — программы, метки — данные. В работе [16] предлагается интерпретация: места — не имеют термина в рамках задачи построения модели процесса, вспомогательные узлы, переходы — действия, метки — прогресс выполнения некоторого исполнения процесса.

2.2 Определения

Далее вводится ряд определений, необходимых для формулировок условий корректности, которым должна удовлетворять модель. Основание для выбора именно таких условий корректности соответствуют результатам, представленным в [40], которые посвящены исследованию о выборе наилучшей модели для описания функционирующей информационной технологии. Определения вводятся по аналогии с работой [16], однако ряд определений были скорректированы в сторону их упрощения. Рассматривается класс сетей Петри, который в литературе обозначается как PT-сеть (Place/Transition net) [16].

Определение 2.1 (Сеть Петри)**.** Сеть Петри N — тройка (P, T, F), где:

1. P — конечное множество, *места*;
2. T — конечное множество, *переходы*, $P \cap T = \varnothing$;

3. $F \subseteq (P \times T) \cup (T \times P)$ — множество направленных некратных дуг, отношение инцидентности.

Для описания динамического поведения сети Петри вводится определение разметки над множеством мест P. Обычно [16] под разметкой понимается мультимножество мест из P. Так, запись $s = \{p_1, p_1, p_2\}$ означала бы, что в сети N есть 3 метки, 2 из них содержатся в месте $p_1 \in P$, одна метка содержится в месте $p_2 \in P$. В рамках диссертации рассматриваются разметки, в которых в каждом месте сети N может находиться не более одной метки.

Определение 2.2 (Размеченная сеть Петри)**.** Пусть $N = (P, T, F)$ — сеть Петри. Разметка s — множество мест из P, в которых есть метка. Пара $(N = (P, T, F), s)$ — размеченная сеть Петри.

Пусть $\mathcal{N}$ — множество всех размеченных сетей Петри.

Определение 2.3 (Входной и выходной узел)**.** Пусть $N = (P, T, F)$ — сеть Петри. Элементы из $P \cup T$ — *узлы*. Пусть $x, y \in P \cup T$.

Узел x — *входной узел* узла y, если существует направленная дуга из x в y, то есть $(x, y) \in F$. Входной узел узла y обозначается следующим образом:

$$\bullet y = \{x | (x, y) \in F\}.$$

Узел y — *выходной узел* узла x, если $(x, y) \in F$. Выходной узел узла x обозначается следующим образом:

$$x\bullet = \{y | (x, y) \in F\}.$$

На представленном ранее Рис. 3 8 мест (круги), 7 переходов (квадраты). Переход A имеет 1 входное место и 1 выходное место, переход *AND-split* имеет одно входное место и 2 выходных. Черный круг во входном месте A — токен, изначальная разметка.

Динамическое поведение сети Петри задается с помощью сформулированного далее правила.

Определение 2.4 (Правило срабатывания переходов)**.** Пусть $(N = (P, T, F), s)$ — размеченная сеть Петри. Переход $t \in T$ *активирован*,

$(N, s)[t\rangle$, если $\bullet t \subseteq s$. Правило срабатывания переходов $\_[\_\rangle\_ \subseteq \mathcal{N} \times T \times \mathcal{N}$ — наименьшее отношение, удовлетворяющее: $\forall (N = (P, T, F), s) \in \mathcal{N}$ и $\forall t \in T$:

$$(N, s)[t\rangle \implies (N, s)[t\rangle(N, s \setminus (\bullet t) \cup (t\bullet))$$

Это выражение означает следующее: сеть Петри N с разметкой s, после срабатывания активированного перехода t, переходит в разметку $s \setminus (\bullet t) \cup (t\bullet)$, то есть множество меток из входных мест для перехода t переходит в выходные места перехода t.

Для примера на Рис. 3: активирован переход A, после срабатывания правила перехода токен удаляется из входного места и помещается в выходное место. В результирующей разметке активированы 2 перехода: E и *AND-split*. Далее может сработать только один из переходов (понятие *конфликта* в [60]). Если срабатывает *AND-split*, токен удаляется из входного места, 2 токена помещаются в выходные места.

Определение 2.5 (Достижимые разметки)**.** Пусть (N, s_0) — размеченная сеть Петри из $\mathcal{N}$. Разметка s *достижима* из начальной разметки s_0, если существует последовательность активированных переходов таких, что их последовательное срабатывание приведут из разметки s_0 в разметку s. Множество достижимых разметок (N, s_0) обозначается $[N, s_0\rangle$.

Определение 2.6 (Срабатывающая последовательность)**.** Пусть (N, s_0), где $N = (P, T, F)$ — размеченная сеть Петри. Последовательность переходов $\sigma \in T^*$ называется *срабатывающей* для (N, s_0), если $\exists n \in \mathbb{N} \cup \{0\}$ такое, что существуют разметки $s_1, \ldots, s_n$ и переходы $t_1, \ldots, t_n \in T$ такие, что $\sigma = t_1 \ldots t_n$ и $\forall i, 0 \leqslant i < n$, $(N, s_i)[t_{i+1}\rangle$ и $s_{i+1} = s_i \setminus (\bullet t_{i+1}) \cup (t_{i+1}\bullet)$.

То есть для последовательности $\sigma = t_1 \ldots t_n$ существуют разметки $s_1, \ldots, s_n$ и переходы $t_1, \ldots, t_n \in T$ такие, что каждый переход t_{i+1}, $0 < i \leqslant n$, активирован в разметке s_i и после срабатывания переводит сеть Петри N в разметку s_{i+1}.

Последовательность σ называется *активированной* в разметке s_0: $(N, s_0)[\sigma\rangle$.

Срабатывание последовательности σ ведет к разметке s_n: $(N, s_0)[\sigma\rangle(N, s_n)$

Если $n = 0$, то $\sigma = \varepsilon$, где ε – пустая последовательность. Таким образом, пустая последовательность также является срабатывающей последовательностью для (N, s_0).

Далее рассматриваются сети Петри с двумя выделенными местами: $i \in P$ —- входное место сети, $o \in P$, $i \neq o$, —- выходное место сети. Предполагается, что в сеть N новая метка может поступать только во входное место и покидать сеть через выходное место сети. Также далее рассматриваются только связные сети Петри (workflow nets [16]).

Пусть R^{-1} — обратное отношение к отношению R, а R^* — рефлексивное и транзитивное замыкание отношения R.

Определение 2.7 (Связная сеть Петри)**.** Сеть Петри $N = (P, T, F)$ *(слабо) связна*, если для любых $x, y \in P \cup T$, справедливо $x(F \cup F^{-1})^* y$. Сеть Петри $N = (P, T, F)$ *сильно-связна*, если для любых $x, y \in P \cup T$, справедливо xF^*y.

Так, пример на Рис. 3 является связной, но не является сильно-связной сетью Петри, так как не существует пути от действия D к действию A.

Далее будут рассматриваться только связные сети Петри, которые имеют не менее двух мест. Несмотря на такое сужение множества сетей Петри, все еще остается актуальным решение задач: исключение зацикливания последовательностей действий в рамках процесса; обнаружения действий, которые никогда не выполнятся; обнаружения и удаления мусора после завершения выполнения объекта.

Пример размеченной сети Петри $(N = (P, T, F), \{i\})$, где $P = \{i, o, p_1, p_2\}$, $T = \{t_1, t_2\}$, $F = \{(i, t_1), (t_1, p_1), (t_1, p_2), (p_1, t_2), (t_2, o)\}$, приведен на Рис. 4. Метка в месте i активирует переход t_1. Множество входящих мест в переход t_1: $(\bullet t_1) = \{i\}$, множество выходящих мест $(t_1 \bullet) = \{p_1, p_2\}$.

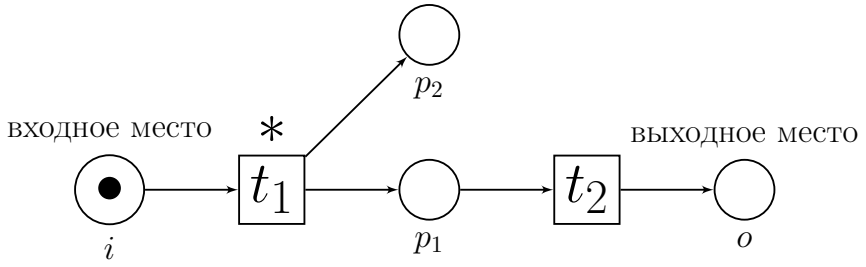

Рис. 4 — Размеченная сеть $(N, \{i\})$. Активированный переход отмечен $*$

Пусть срабатывает переход t_1. Следует отметить, что если в сети Петри есть несколько активированных переходов, то в следующий момент времени может сработать любое подмножество из них [17]. В диссертации предполагается, что в один момент времени может сработать только 1 из активированных переходов. По правилу срабатывания переходов, новая разметка для сети N после срабатывания перехода t_1: $s \cup (t\bullet) \setminus (\bullet t) = \{i\} \cup \{p_1,p_2\} \setminus \{i\} = \{p_1,p_2\}$, становится активированным переход t_2, Рис. 5.

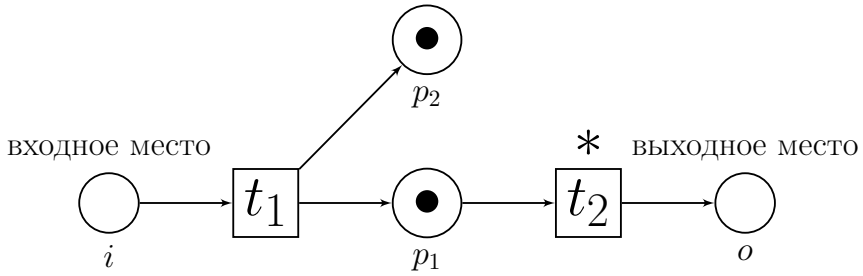

Рис. 5 — Размеченная сеть $(N,\{p_1,p_2\})$

Пусть срабатывает переход t_2. Множество входящих мест в переход t_2: $(\bullet t_2) = \{p_1\}$, множество выходящих мест $(t_2\bullet) = \{o\}$. Новая разметка после срабатывания перехода t_2 по правилу срабатывания переходов: $\{p_1,p_2\} \cup \{o\} \setminus \{p_1\} = \{p_2,o\}$, Рис. 6.

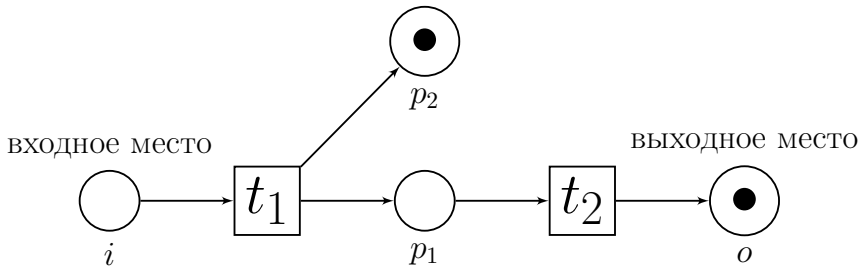

Рис. 6 — Размеченная сеть $(N,\{p_2,o\})$

Определение 2.8 (Замкнутость и безопасность)**.** Размеченная сеть $(N = (P,T,F),s)$ *замкнута*, если множество достижимых разметок $[N,s\rangle$ конечно. Размеченная сеть *безопасна*, если $\forall s' \in [N,s\rangle$ и $\forall p \in P$, $s'(p) \leqslant 1$, где $s'(p)$ – количество меток в разметке s' в месте p.

Свойства замкнутости и безопасности налагают ограничения на рассматриваемый класс сетей Петри, ограничивая его сетями Петри, которые имели бы конечное множество возможных разметок таких, что в каждом

месте сети Петри содержалось бы не более одной метки в один момент времени. Справедливо следующее утверждение:

Утверждение 2.1. *Условие безопасности влечет условие замкнутости.*

Доказательство. Пусть есть безопасная размеченная сеть $(N = (P, T, F), s)$.

Условие безопасности означает, что для всех достижимых разметок сети N из разметки s, для всех мест, количество меток в месте равно 0 или 1.

Если $|P| = n$, и в каждом месте может быть не более 1 метки, то количество различных разметок не превышает 2^n. Следовательно, по определению, сеть $(N = (P, T, F), s)$ является замкнутой. ■

Определение 2.9 (Мертвые переходы, жизнеспособность)**.** Пусть $(N = (P, T, F), s)$ — размеченная сеть Петри. Переход $t \in T$ *мертвый*, если не существует достижимой разметки $s' \in [N, s\rangle$ такой, что $(N, s')[t\rangle$.

Сеть (N, s) — *жизнеспособна*, если для всех достижимых разметок $s' \in [N, s\rangle$, $\forall t \in T$, существует достижимая разметка $s'' \in [N, s'\rangle$ такая, что $(N, s'')[t\rangle$.

То есть сеть Петри является жизнеспособной, если для всех достижимых разметок s' и для всех переходов t существует достижимая из s' разметка s'' такая, что она активирует переход t. Если сеть жизнеспособа, то в сети отсутствуют мертвые переходы. Более того, условие жизнеспособности соответствует определению L_4-live, а условие отсутствия мертвых переходов соответствует определению L_0-live определений возможных степеней жизнеспособности сетей Петри, рассматриваемых, например, в работе [62]. Так, на Рис. 3 в сети Петри нет мертвых переходов.

Сужение класса сетей Петри до сетей Петри без мертвых переходов позволит в дальнейшем строить конечные и неизбыточные модели процесса по его логу.

Рассматриваются свойства корректности, налагаемые на модель, представленную сетью Петри. Выбор именно этих свойств корректности мотивирован их простотой и предположением, что наложение таких свойств

корректности на модель позволит получить более надежные результаты при последующем решении задачи поиска аномалий.

Первое условие корректности формулируется следующим образом.

Определение 2.10 (Свойство надежности (Sound)). Пусть $N = (P, T, F)$ — сеть Петри с входным местом i и выходным местом o. N *надежна*, если справедливы следующие утверждения.

1. $(N, [i])$ безопасна.
2. $(N, [i])$ корректно завершается: $\forall s \in [N, [i]\rangle$, $o \in s \implies s = \{o\}$.
3. $(N, [i])$ не содержит мертвых переходов.

Можно видеть, что такое условие корректности налагает минимальные интуитивные необходимые ограничения на сеть Петри для возможности дальнейшего поиска аномалий. Свойство безопасности отвечает за наличие не более одной метки в одном месте сети Петри в один момент времени; свойство корректного завершения говорит о том, что если метка попала в выходное место, эта метка должна быть единственной в сети Петри; свойство отсутствия мертвых переходов говорит о том, что в сети Петри не должно быть переходов, которые не активируются ни в одной из достижимых разметок.

Определение 2.11 (Действие, трасса, лог). Пусть V — конечное непустое множество действий. $\omega \in V^*$ — *трасса*. $L \subseteq V^*$ — неупорядоченное конечное множество трасс, *лог*.

Определение 2.12 (Отношение порядка между действиями в логе). Пусть L — лог над V. Пусть $A, B \in V$:

- $A >_L B$, если $\exists \omega = C_1 \dots C_n$, $C_i \in V$, $i \in \{1, \dots, n-1\}$ такие, что $\omega \in L$ и $C_1 = A$, $C_{i+1} = B$ — отношение предшествия, A *предшествует* B;
- $A \to_L B$, если $A >_L B$ и $B \not>_L A$ — прямое каузальное отношение;
- $A \#_L B$, если $A \not>_L B$ и $B \not>_L A$ — действия не встречаются вместе;
- $A ||_L B$, если $A >_L B$ и $B >_L A$ — потенциальный параллелизм.

Утверждение 2.2. *(Van der Aalst [16]) Пусть L лог над V. Для всех $A, B \in V$ выполнено $A \to_L B$, или $B \to_L A$, или $A \#_L B$, или $A ||_L B$. При*

этом множества, соответствующие этим бинарным отношениям, не пересекаются и их объединение составляет $V \times V$.

Доказательство. Доказательство в работе [16] не приведено, но его можно провести следующим образом.

Если переобозначить множества, соответствующие бинарным отношениям: $>_L = Q$, $>_L^{-1} = R$, $\rightarrow_L = S$, $\rightarrow_L^{-1} = D$, $\#_L = E$, $||_L = F$. Тогда, согласно определению 2.12:

$$S = Q \backslash R$$

$$D = R \backslash Q$$

$$E = (V \times V) \backslash (Q \cup R)$$

$$F = Q \cap R$$

Очевидно, что множества S, D, E, F не пересекаются. Если рассмотреть объединение $S \cup D \cup E \cup F$:

$$(Q \backslash R) \cup (R \backslash Q) \cup ((V \times V) \backslash (Q \cup R)) \cup (Q \cap R)$$

Так как $(Q \backslash R) \cup (R \backslash Q) \cup (Q \cap R) = R \cup Q$:

$$S \cup D \cup E \cup F = V \times V.$$

■

Определение 2.13 (Связь между переходами сети Петри и действиями лога)**.** Связь между переходами сети Петри $N = (P, T, F)$ и действиями лога $L \subseteq V^*$ задается сюръективным отображением $\tau : T \rightarrow V$.

Традиционно предполагается [16], что отображение τ является биективным, однако, в рамках диссертации рассматриваются лог и более широкий класс сетей Петри такие, что $|T| \geqslant |V|$.

Определение 2.14 (Свойство полноты лога)**.** Пусть $N = (P, T, F)$ — надежная сеть Петри, L — лог над V. Отображение $\tau : T \rightarrow V$ задает связь между переходами сети Петри N и действиями лога L.

L — *лог сети* N, если для каждой трассы лога $\omega = \tau(t_1), \ldots, \tau(t_n) \in L$, соответствующая последовательность переходов $\sigma = t_1, \ldots, t_n$ — является срабатывающей последовательностью в N, начиная с разметки $[i]$ и заканчивая разметкой $[o]$. То есть $(N, [i])[\sigma\rangle(N, [o])$.

L — *полный лог сети* N, если:

1. L — лог рабочего процесса;
2. для любого другого лога рабочего процесса L' сети N выполнено: $>_{L'} \subseteq >_L$;
3. $\forall t \in T\ \exists \omega \in L$: $\tau(t) \in \omega$ — все переходы покрываются некоторой срабатывающей последовательностью.

Свойство полноты лога процесса описывает следующее условие корректности. Это условие корректности описывает связь между логом и сетью Петри, в частности, сможет ли построенная сеть Петри породить тот же лог, из которого данная сеть была построена. Если построенная модель описывает более широкий класс трасс, часть которых не описывает легальное поведение системы, то задача поиска аномалий не может быть корректно сформулирована, так как теряется возможность отличить легальные трассы от аномальных.

Основной трудностью в формировании полного лога является то обстоятельство, что если модели присущ параллелизм или циклы, то количество трасс, которые могут быть порождены моделью, может быть бесконечно. Даже если модель не содержит циклов и параллелизма, но в ней есть N бинарных условных переходов, то количество трасс для формирования полного лога в худшем случае может составить 2^N. Более того, маловероятно, что для построения модели использовался полный лог. Это означает, что идеальным случаем является тот случай, при котором: есть доступ ко всем возможным трассам процесса, тем самым есть возможность порождать полный лог; строить по этому логу модель, порождать построенной моделью трассы, типичные для процесса.

Следующая далее Теорема 2.1 констатирует следующее: если существует каузальное отношение между действиями согласно логу рабочего процесса, то в модели (сети Петри) существует место, соединяющее перехо-

ды, соответствующие этим двум действиям. Для утверждений работы [16] далее предполагается, что τ — тождественное отображение.

Теорема 2.1. *(Van der Aalst [16]) Пусть $N = (P, T, F)$ — надежная сеть Петри и L — полный лог N. Для всех $a, b \in T$ таких, что $a \to b$ следует $a\bullet \cap \bullet b \neq \varnothing$.*

Данная теорема описывает компромисс между возможностью сети Петри описывать весь лог и при этом иметь минимально возможную сложность, с точки зрения количества структурных элементов в этой сети.

Определение 2.15 (Неявное место)**.** Пусть $N = (P,T,F)$ — сеть Петри с начальной разметкой s. Место $p \in P$ называется *неявным*, если для всех достижимых разметок $s' \in [N, s\rangle$ и переходов $t \in p\bullet$ выполнено: если $s' \supseteq \bullet t \backslash \{p\}$, то $s' \supseteq \bullet t$.

То есть место p будет неявным, если для каждой достижимой из s разметки s' и переходов t, которые следуют после этого места p, справедливо: если в s' есть метки в местах, предшествующих переходу t, то в месте p в этой разметке тоже будет метка.

Утверждается [16], что никакой алгоритм построения модели процесса не сможет обнаружить неявные места, так как эти места не влияют на поведение сети и не видны в логе. Поэтому далее рассматриваются сети без неявных мест.

Пример на Рис. 3 не содержит неявных мест. Например, если добавить место, соединяющее переходы A и D, это место будет неявным. Место было бы явным, если в логе в Таблице 1 существовала бы трасса AD.

Свойство корректности заключается в отсутствии запрещенных конструкций, такие конструкции показаны на Рис. 7. Эти конструкции соответствуют ситуациям, когда происходят подряд выбор и синхронизация. Было показано [40], что наличие таких конструкций не только ухудшает восприятие человеком модели, но и усложняет процесс отладки модели, а следовательно, делает более сложным последующее решение задачи поиска аномалий. Под *выбором* в контексте сетей Петри понимается конструкция, когда из одного места существуют ребра в несколько переходов. Под *синхронизацией* — когда из нескольких мест есть ребра в

один и тот же переход. Таким образом, левый рисунок на Рис. 7 описывает ситуацию «синхронизация и выбор»: ситуация выбора действия (место с многими выходными переходами) не должна происходить одновременно с синхронизацией действий (переходы с многими входными местами). Правый рисунок на Рис. 7 описывает ситуацию “выбор и синхронизация”: если происходит синхронизация действий (переход с многими входными местами), то все предшествующие действия должны быть выполнены. То есть синхронизация действий не должна происходить сразу после операции “ИЛИ”.

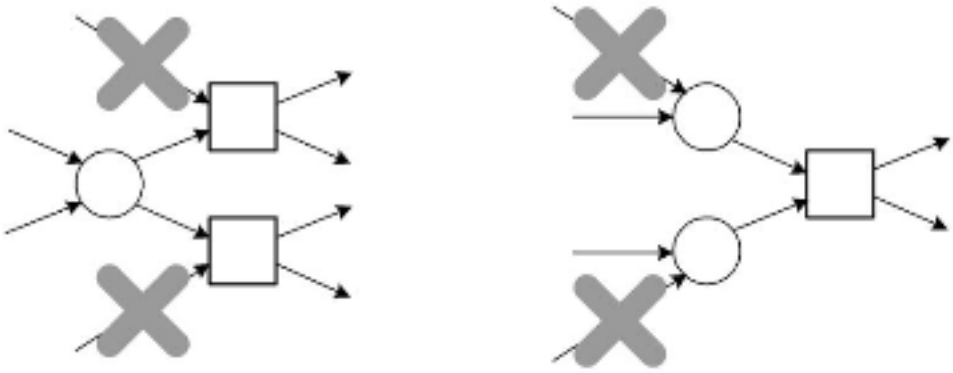

Рис. 7 — Запрещенные конструкции в сетях Петри

Таким образом, еще больше сужается класс подходящих сетей Петри:

Определение 2.16 (Структурированная сеть Петри (SWF-сеть))**.** Сеть Петри $N = (P, T, F)$ — SWF-сеть, если $\forall p \in P, \forall t \in T$, где $(p,t) \in F$, выполнено:

1. если $|p \bullet| > 1$, то $|\bullet\, t| = 1$ (левый рисунок на Рис. 7);
2. если $|\bullet\, t| > 1$, то $|\bullet\, p| = 1$ (правый рисунок на Рис. 7);
3. в сети нет неявных мест.

SWF-сети являются подклассом сетей свободного выбора [63]. Для таких сетей существует алгоритм, определяющий, надежна ли сеть, и работающий за полиномиальное время [64]. Для SWF-сетей также справедливо утверждение, сформулированное в [16] и с помощью которого возможно идентифицировать одинаковые места.

Утверждение 2.3 (Van der Aalst [16]). *Пусть $N = (P, T, F)$ — SWF-сеть. Для всех $a, b \in T$ и $p_1, p_2 \in P$ выполнено: если $p_1 \in a\bullet \cap \bullet b$ и $p_2 \in a\bullet \cap \bullet b$, то $p_1 = p_2$.*

Утверждение следует из того, что по определению SWF-сети никакие 2 перехода не соединены несколькими местами. Для SWF-сетей справедливо обратное утверждение Теоремы 2.1, представленной в [16].

Утверждение 2.4 ([16]). *Пусть $N = (P, T, F)$ — надежная SWF-сеть, L — полный лог рабочего процесса сети N. Для всех $a, b \in T$ выполнено: если $a\bullet \cap \bullet b \neq \varnothing$, то $a >_L b$.*

На Рис. 8 представлены примеры сетей Петри, для которых работает предлагаемый в [16] алгоритм майнинга.

В работе [16] вводится ряд свойств корректности, налагаемых на сеть Петри для того, чтобы модель могла быть построена по логу рабочего процесса. Данные свойства согласуются с исследованием [40], которое посвящено изучению вопроса выбора наилучшей модели процесса, исходя из интуитивных предположений и эмпирических наблюдений. Мотивацией выбора наилучшей модели является удобство поиска ошибок в самой модели, ее простота и способность оставаться понятной конечному пользователю с течением времени. Данные аспекты также актуальны для эффективного решения задачи поиска аномалий, так как напрямую влияют на то, насколько возможно доверять модели, которая описывала бы легальные исполнения процесса. Далее в работе представлен анализ совместимости следующих 4 свойств корректности:

1. свойство надежности, определение 2.10;
2. свойство запрета последовательного выполнения в сети конструкций синхронизации и выбора, определение 2.16;
3. полнота лога рабочего процесса для рассматриваемой сети, определение 2.14;
4. сохранение в сети отношения каузальности между переходами, если между соответствующими действиями в логе это отношение было выполнено, Теорема 2.1.

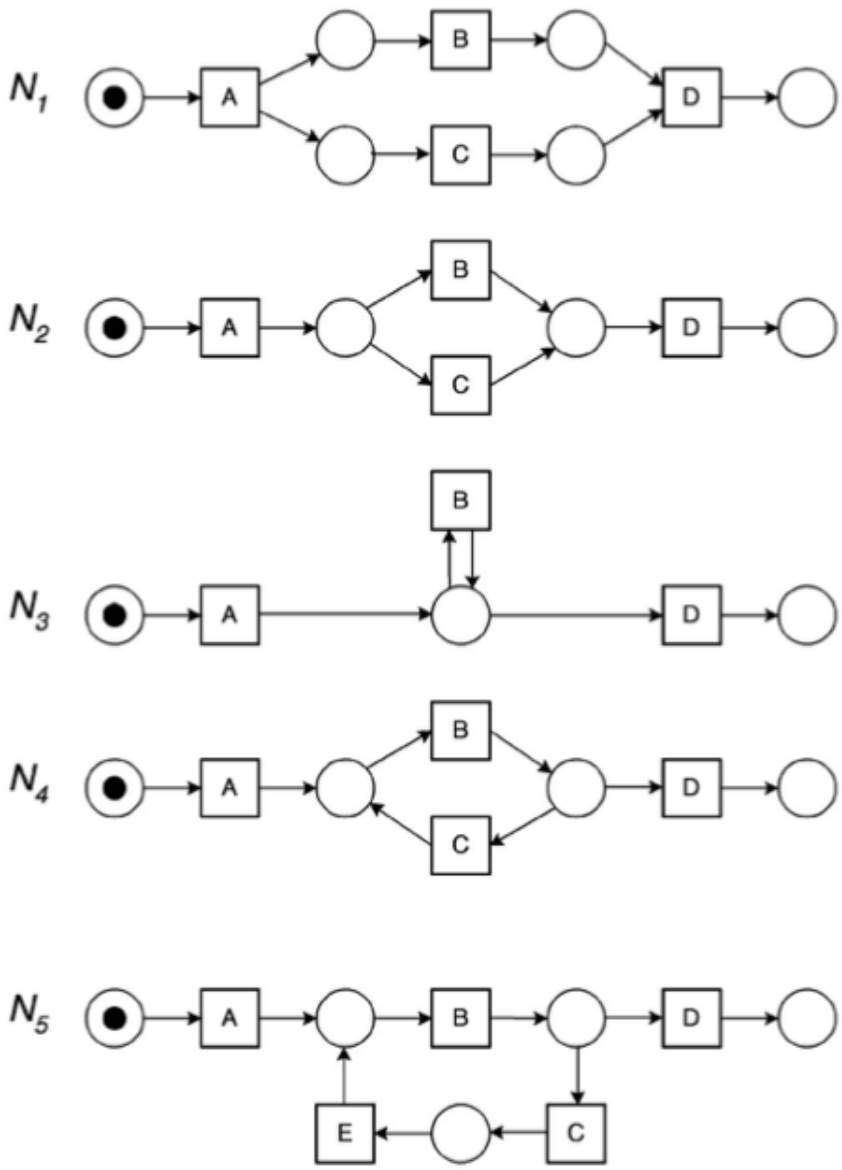

Рис. 8 — Примеры надежных SWF-сетей

2.3 Задача построения модели процесса по логу с наложением условий корректности

Рассматривается постановка задачи построения модели рабочего процесса, который представлен логом $\mathbb{L}$. Если такая модель рабочего процесса построена, то для нее применим алгоритм построения процесса, предложенный в работе [16]. В настоящей диссертации приведен пример рабочего процесса, состоящего из 4 действий, для которого, в терминах предло-

женного в работе [16] математического аппарата, не удается построить корректную модель, если под корректностью понимается выполнение 4 свойств, описанных ранее.

Описание процесса

Пусть рассматривается некоторое простейшее многопользовательское приложение, легальным поведением в котором является: пользователь авторизуется в своем профиле (действие `login`), возможно, не с первой попытки (2 подряд действия `login` являются типичными); совершает или не совершает допустимое действие приложения (действие `post personal` или ничего); сохраняет изменения и выходит из своего профиля (действия `save profile` и `logout`), Рис. 9.

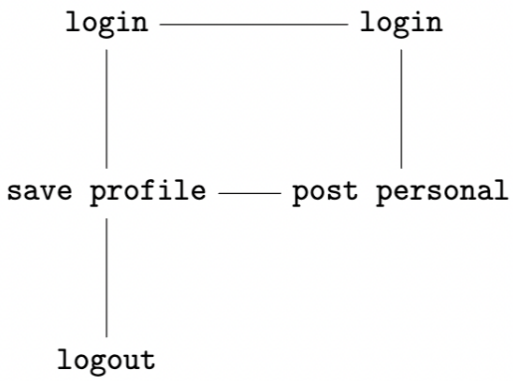

Рис. 9 — Диаграмма Хассе многопользовательского приложения

Если переименовать действия, то, в настоящей главе рассматривается процесс, состоящий из 4 действий:

- действие A, которое всегда выполняется в начальный момент времени;
- блок “ИЛИ”, выбор: либо действие A, либо действие B, либо никакого действия;
- действия C и D, которые всегда заключают исполнение процесса.

Полный лог для такого процесса выглядит следующим образом: $L = \{ACD, AACD, ABCD\}$. Для лога L справедливы следующие отношения:

- отношение предшествия между действиями (переходами в сети Петри): $A >_{\mathbb{L}} A, A >_L B, A >_L C, B >_L C, C >_L D$;
- прямое каузальное отношение $A \to_L B$, $A \to_L C$, $B \to_L C$, $C \to_L D$;
- отношение параллелизма $A||_L A$;
- действия не встречаются вместе (симметричное отношение, для краткости не приведены записи вида bRa, если есть запись aRb): $A\#_L D$, $B\#_L B$, $B\#_L D$, $C\#_L C$, $D\#_L D$.

Для данного лога, описывающего процесс, возможно построить несколько моделей в виде сетей Петри. Далее рассматриваются модели процесса и анализируется совместимость свойств корректности, описанных выше.

Рассматриваются сети Петри для полного лога L такие, что сеть Петри удовлетворяла бы определению надежности и была бы SWF-сетью. В таком случае можно было бы утверждать, что построенная сеть может быть переобнаружена α-алгоритмом из работы [16]. Сети Петри, представленные в данном разделе, могут рассматриваться, как потенциальные модели процесса, так как трассы лога L могут быть порождены этими сетями. Или, другими словами, сети $\mathbb{N}_1$, $\mathbb{N}_3$, $\mathbb{N}_4$, $\mathbb{N}_5$ моделируют лог L. Сеть $\mathbb{N}_2$ представлена как пример неверной работы сложных блоков *OR-split*, *OR-join* в классе сетей Петри, предлагаемых авторами [16].

Сеть Петри $\mathbb{N}_1$ для лога L

Рассматривается сеть Петри на Рис. 10 на соответствие логу L. Данная сеть Петри полностью соответствует определениям работы [16], то есть связь между переходами сети Петри и действиями лога задается с помощью тождественного отображения τ.

Можно видеть, что в сети на Рис. 10, помимо последовательностей из лога L, возможны, например, цепочки:

```
A AA...AA CD
A AB...AB CD
A BB...BB CD
```

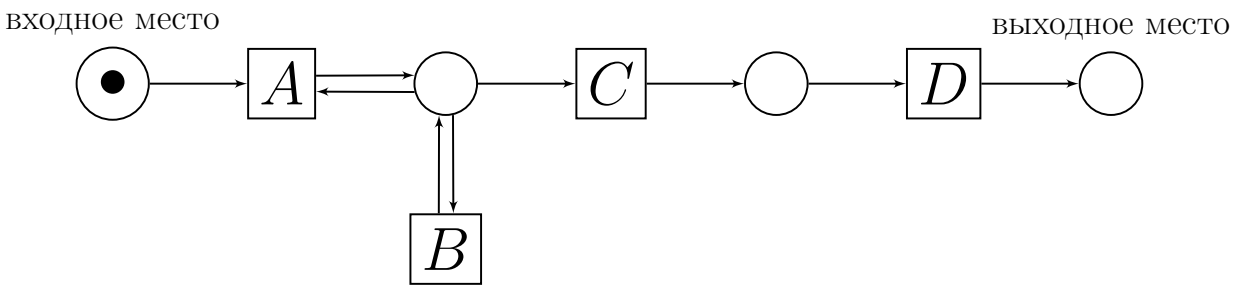

Рис. 10 — Сеть Петри N_1 для лога L

Справедливо следующее утверждение.

Лемма 2.1. *Сеть* N_1 *надежна.*

Доказательство. Для доказательства утверждения необходимо проверить выполнение пункты определения 2.10.

– Проверка безопасности: в каждый момент времени в каждом месте должны находиться 0 или 1 метка.

  Пусть в изначальной разметке во входном месте находится одна метка (токен) (как на Рис. 10).

  В сети N_1 нет переходов, у которых несколько выходных мест (из которых исходит более одного ребра), поэтому количество токенов не может увеличиться. Таким образом, сеть безопасна.

– Проверка корректного завершения: $\forall s \in [\mathrm{N}_1, [i]\rangle$, $o \in s \implies s = [o]$. Где $[i]$ — разметка, в которой есть один токен и он находится во входном месте, $[o]$ — разметка, в которой есть один токен и он находится в выходном месте.

  То есть корректное завершение означает, что если исполнение процесса началось с разметки, где был только один токен, находящийся во входном месте, и в какой-то момент токен окажется в выходном месте, то этот токен будет единственным.

Так как сеть $\mathbb{N}_1$ безопасна и в сети нет переходов, увеличивающих количество меток, то если в разметке $[i]$ был 1 токен, то если этот токен дойдет до выходного места, он будет единственным. Таким образом, сеть $\mathbb{N}_1$ корректно завершается.

– Проверка отсутствия мертвых переходов.

  По определению: переход $t \in T$ *мертвый*, если не существует достижимой разметки $s' \in [\mathbb{N}_1, s\rangle$ такой, что $(\mathbb{N}_1, s')[t\rangle$.

  Для доказательства отсутствия мертвых переходов рассматривается последовательность переходов $\sigma = t_1, t_2, t_3$. Данная последовательность активирует все переходы сети и переводит метку из входного места в выходное место — $(\mathbb{N}_1, [i])[\sigma\rangle(\mathbb{N}_1, [o])$, то есть является срабатывающей последовательностью для сети $\mathbb{N}_1$.

  – Разметка $[i]$ активирует переход A, Рис. 11.

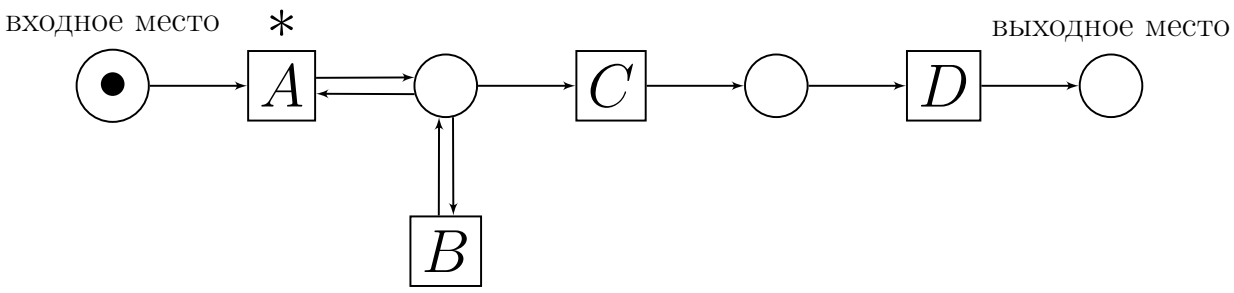

Рис. 11 — Часть сети $\mathbb{N}_1$ в изначальной разметке, активированный переход помечен $*$

  – $t_1 = A$, при срабатывании токен перемещается в место между переходами A, B и C, активируются переходы A, B и C, Рис. 12.

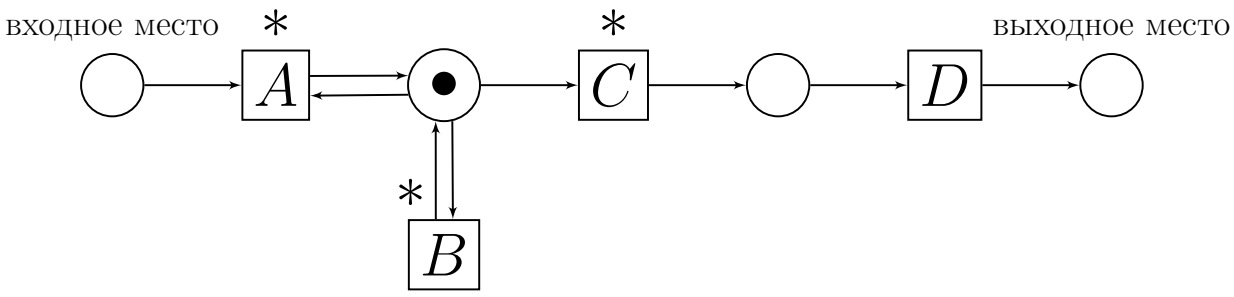

Рис. 12 — Часть сети $\mathbb{N}_1$ после срабатывания t_1

– $t_2 = B$, токен остается в том же месте, активированы те же 3 перехода, Рис. 12.
– $t_2 = C$, токен перемещается в место между переходами C и D, активирован переход D, Рис. 13.

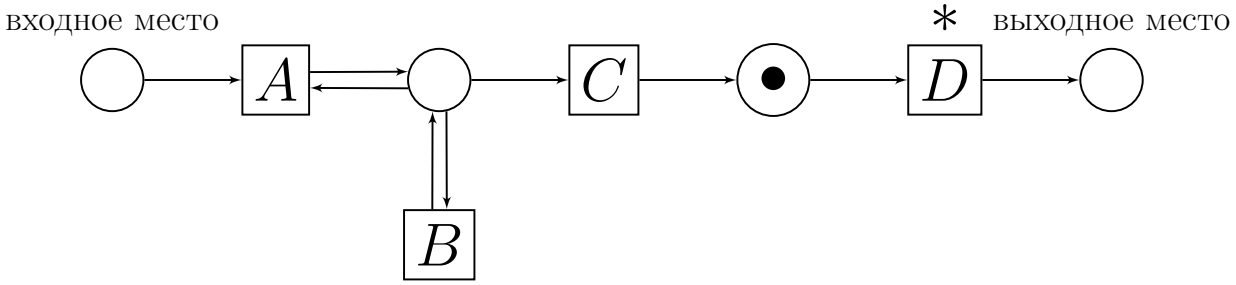

Рис. 13 — Часть сети $\mathbb{N}_1$ после срабатывания t_2

– $t_3 = D$, токен перемещается в выходное место, Рис. 14.

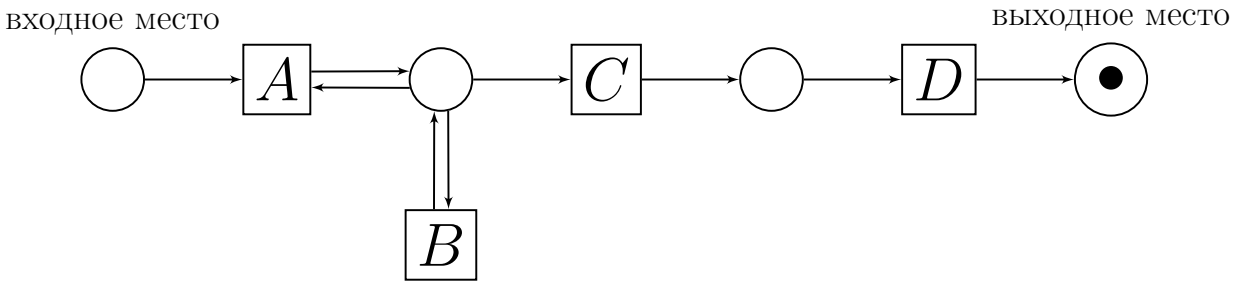

Рис. 14 — Часть сети $\mathbb{N}_1$ после срабатывания t_3

Таким образом, на Рис. 11-14 показаны достижимые разметки из разметки $[i]$, в которых активируются все переходы сети $\mathbb{N}_1$, то есть в сети $\mathbb{N}_1$ нет мертвых переходов.

Выполнены все свойства определения надежности, поэтому сеть $\mathbb{N}_1$ надежна. ■

Сеть $\mathbb{N}_1$ соответствует Теореме 2.1, так как для каждой пары действий, между которыми есть каузальное отношение, в сети $\mathbb{N}_1$ между соответствующими переходами есть место.

Построенная сеть $\mathbb{N}_1$ (Рис. 10) не является SWF-сетью, так как содержит обе запрещенные конструкции из Рис. 7. Далее на Рис. 15 и 16 ребра сети, не участвующие в демонстрации нарушения, сделаны более тонкими,

а ребра, демонстрирующие нарушение условия, наоборот, сделаны более толстыми:

1. нарушение условия $\forall(p,t) \in F$, *если* $|p\bullet| > 1$, *то* $|\bullet t| = 1$, Рис. 15;

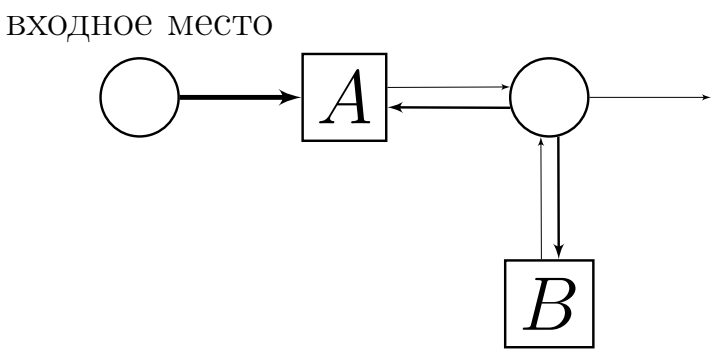

Рис. 15 — Левая запрещенная конструкция в сети $\mathbb{N}_1$

2. нарушение условия $\forall(p,t) \in F$, *если* $|\bullet t| > 1$, *то* $|\bullet p| = 1$, Рис. 16.

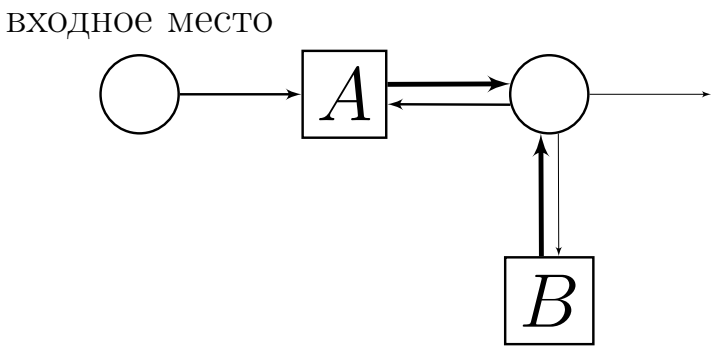

Рис. 16 — Правая запрещенная конструкция в сети $\mathbb{N}_1$

Так как сеть моделирует большее количество трасс, чем содержится в логе L, то не выполняется Свойство 3 о полноте лога.

В сети $\mathbb{N}_1$ выполняется Свойство 4, так как для каждой пары действий в логе L, между которыми есть каузальное отношение, в сети $\mathbb{N}_1$ между соответствующими переходами существует место.

Можно видеть, что в данном классе сетей Петри уже для такого простого лога не удается добиться одновременного выполнения рассматриваемых свойств корректности. Следовательно, для данного примера не удается однозначно восстановить модель процесса. А без построенной корректной модели, описывающей процесс, невозможно решать задачу поиска аномалий.

Далее для лога L будут рассматриваться сети Петри из более широкого класса, где связь между переходами сети Петри и действиями лога задается нетождественным отображением $\tau : T \to V$. Таким образом, не будут выполняться условия Теоремы 2.1, однако будет показано, что в более широком классе сетей Петри для лога L возможно добиться выполнения других свойств корректности.

Сеть Петри $\mathbb{N}_2$ для лога L

Авторы [16] полагают, что SWF-сети поддерживают сложные блоки вида *OR-split*, *OR-join*, *AND-split*, *AND-join*. Так как в логе L есть конструкция "ИЛИ", построим сеть Петри $\mathbb{N}_2$ с переходами *OR-split* и *OR-join*, Рис. 17.

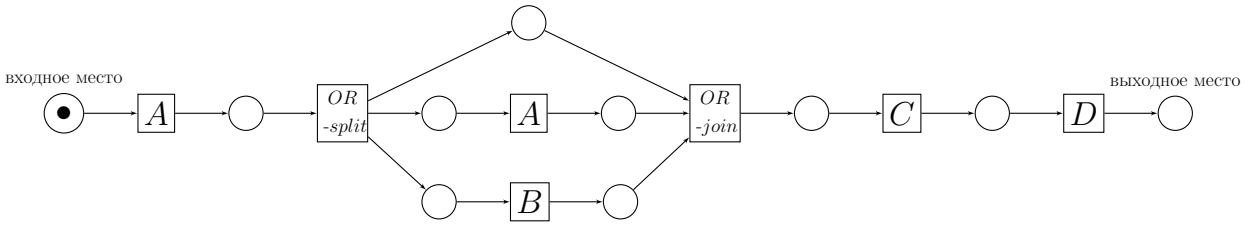

Рис. 17 — Сеть Петри $\mathbb{N}_2$ для лога L

В рассматриваемом классе сетей Петри условия "И" и "ИЛИ" моделируются с помощью следующих конструкций [65], Рис. 18 и Рис. 19.

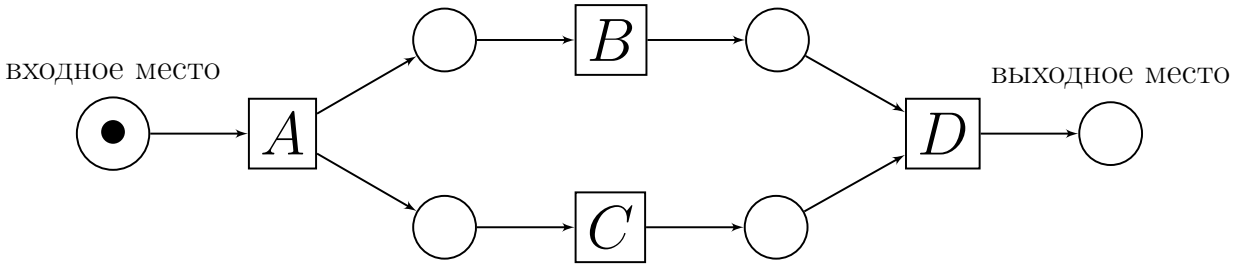

Рис. 18 — Конструкция параллельного исполнения (условие "И")

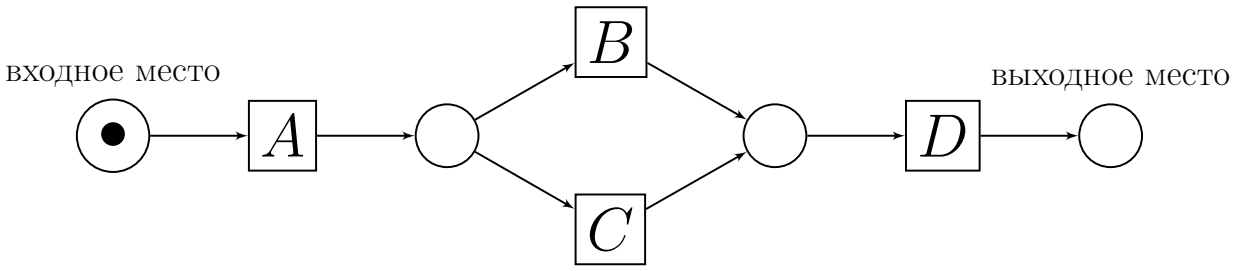

Рис. 19 — Конструкция “race condition” (условие “ИЛИ”)

То есть без наложения дополнительных условий на переходы *OR-split* и *OR-join* в сети на Рис. 17, эти переходы моделируют условие “И”, а не условие “ИЛИ”. Таким образом, сеть N_2 моделирует не лог L, а лог L':

```
A AB CD
A BA CD
```

Выходом было бы рассматривать более сложный класс сетей Петри с наложением условий на переходы, например, класс ингибиторных сетей Петри [61].

Рассматривается полный лог L' для сети N_2. В сети L' есть сложные переходы *OR-split*, *OR-join* и условия Теоремы 2.1 не выполняются. Данная Теорема говорит о том, что если между парой действий справедливо каузальное отношение, то между соответствующими переходами есть общее место, но это не так:

- лог L' не подразумевает наличия дополнительных переходов, кроме указанных в логике процесса действий, в частности действий *OR-split*, *OR-join*, откуда не может быть определено более полное каузальное отношение $\rightarrow$ (чтобы появились конструкции вида $A \rightarrow$ *OR-join*, $B \rightarrow$ *OR-join*);
- в логе L' существует каузальное отношение вида $B \rightarrow_{L'} C$, но при этом в сети N_2, в срабатывающих разметках, токен может находиться либо между переходами B и *OR-join*, Рис. 20, в таком случае, $\bullet C = \varnothing$, откуда $B\bullet \cap \bullet C = \varnothing$. Либо токен может находиться между переходами *OR-join* и C, Рис. 21, в таком случае, $B\bullet = \varnothing$, откуда $B \bullet \cap \bullet C = \varnothing$.

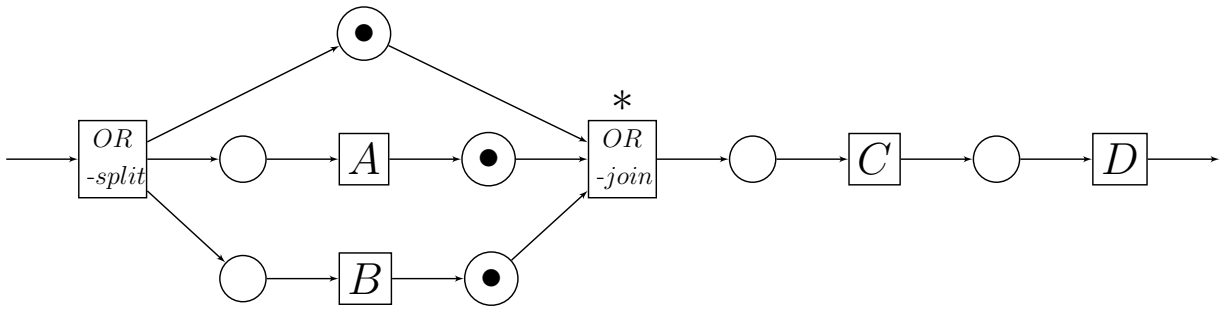

Рис. 20 — Часть сети $\mathbb{N}_2$ после срабатывания последовательности AAB или ABA

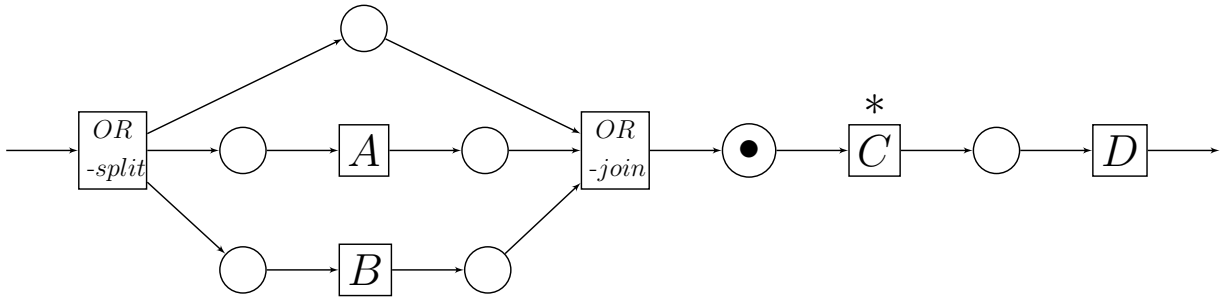

Рис. 21 — Часть сети $\mathbb{N}_2$ после срабатывания последовательности $AABC$ или $ABAC$

Сеть Петри $\mathbb{N}_3$ для лога L

Используя конструкцию “ИЛИ”, Рис. 19, возможно построить сеть Петри $\mathbb{N}_3$ без переходов *OR-split* и *OR-join*, Рис. 22. Основной проблемой остается моделирование пустого перехода в условии “ИЛИ”, так как места в сетях Петри не могут быть соединены с местами.

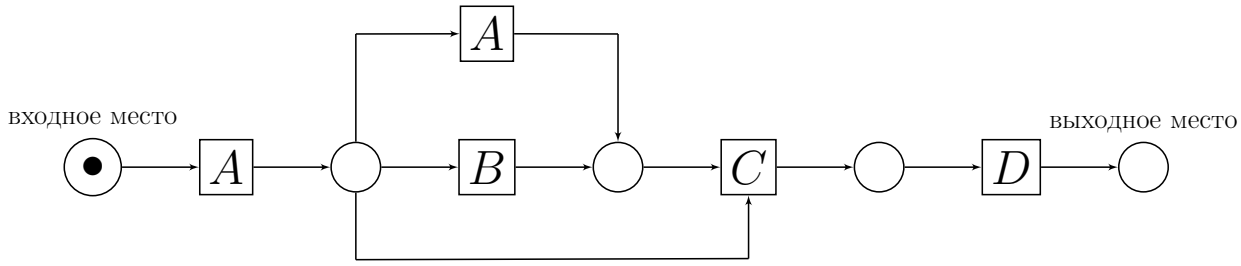

Рис. 22 — Сеть Петри $\mathbb{N}_3$ для лога L

Пусть дана сеть $\mathbb{N}_3$, Рис. 22. Анализ сети:

Сеть $\mathbb{N}_3$ имеет 2 типа переходов:

1. принимает 1 метку, порождает одну метку (например, переход D, переходы с действием A);
2. принимает несколько меток, порождает 1 метку (переход C).

Лемма 2.2. *Сеть* $\mathbb{N}_3$ *надежна.*

Доказательство. Доказательство аналогично доказательству для сети $\mathbb{N}_1$.

Проверка свойств по определению надежности.

– Проверка безопасности: то есть в каждый момент времени в каждом месте должны находиться 0 или 1 метка.

  Пусть в изначальной разметке во входном месте находится одна метка (токен) (как на Рис. 22). В сети $\mathbb{N}_3$ есть 2 типа переходов, как было указано выше: 1 тип не может увеличить количество токенов, 2 тип тоже не может увеличить количество токенов. Поэтому в каждый момент времени в сети в каждом месте будет не более 1 токена. Сеть $\mathbb{N}_3$ безопасна.

  Стоит заметить, что так как в сети нет переходов, которые бы увеличивали количество токенов, второй тип переходов работает точно так же, как первый.

– Проверка корректного завершения: $\forall s \in [\mathbb{N}_3, [i]\rangle,\ o \in s \implies s = [o]$.

  Так как сеть $\mathbb{N}_3$ безопасна, в ней есть только тип переходов, где переход может принять 1 токен и породить 1 токен, и если в сеть Петри не может поступать новый токен (поступление токена во входное место сети), пока находящийся в ней токен не дойдет до выходного места, то если токен окажется в выходном месте, он будет единственным. Таким образом, сеть $\mathbb{N}_3$ корректно завершается;

– Проверка отсутствия мертвых переходов.

  По определению: переход $t \in T$ *мертвый*, если не существует достижимой разметки $s' \in [\mathbb{N}_3, s\rangle$ такой, что $(\mathbb{N}_3, s')[t\rangle$.

Рассматривается срабатывающая последовательность переходов $\sigma = t_1 t_2 t_3 t_4$, активирующая все переходы, переводящую метку из входного места в выходное место — $(\mathbb{N}_3, [i])[\sigma\rangle(\mathbb{N}_3, [o])$.

– Разметка $[i]$ активирует переход A, Рис. 23.

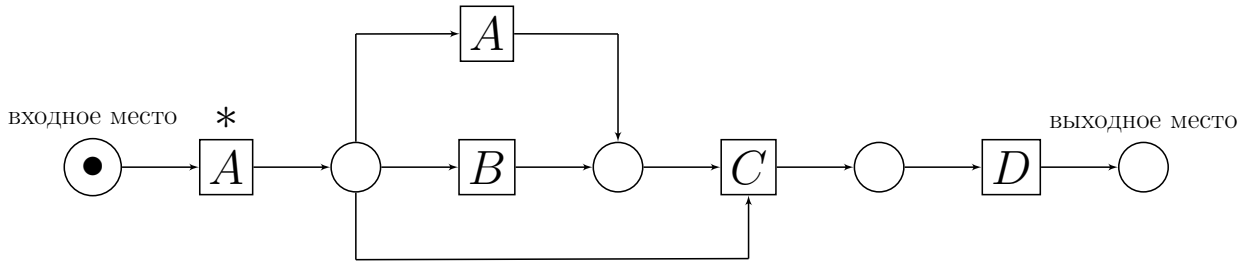

Рис. 23 — Часть сети $\mathbb{N}_3$ в изначальной разметке, активированный переход помечен $*$

– $t_1 = A$, при срабатывании, активируются переходы A, B, C, Рис. 24.

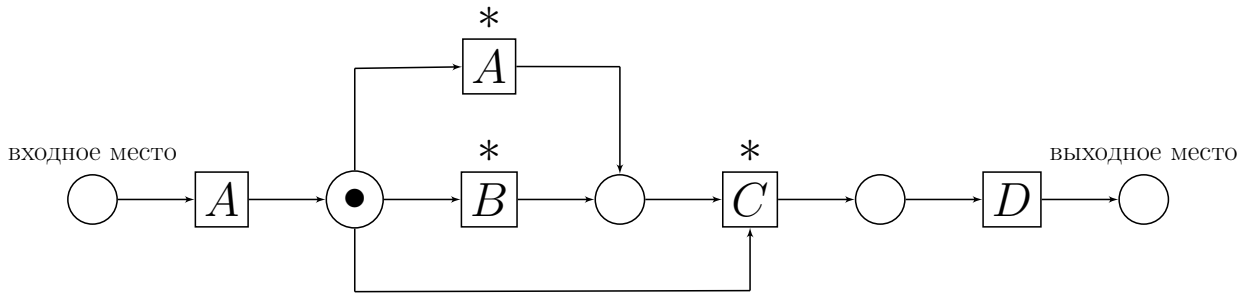

Рис. 24 — Часть сети $\mathbb{N}_3$ после срабатывания t_1

– Моделирование условия “ИЛИ”. Может сработать любой из трех переходов, возможны 3 ситуации:

1. $t_2 = A$, остается активированным 1 переход, переход с действием B не может оставаться активированным, так как токена перед ним больше нет, Рис. 25;
2. $t_2 = B$, переход с действием A не может оставаться актированным, так как токена перед ним больше нет, остается активирован 1 переход, Рис. 25;
3. $t_2 = C$, произойдет ситуация аналогичная далее рассматриваемому переходу t_3, Рис. 26.

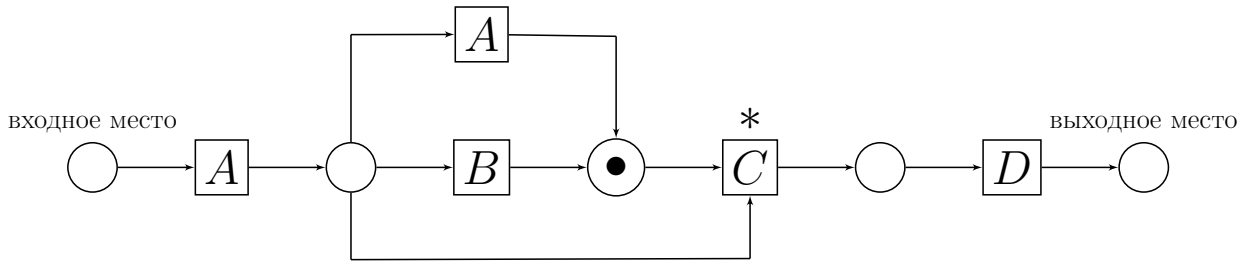

Рис. 25 — Часть сети $\mathbb{N}_3$ после срабатывания $t_2 = A$ или $t_2 = B$

– $t_3 = C$, активируется переход D. Рис. 26.

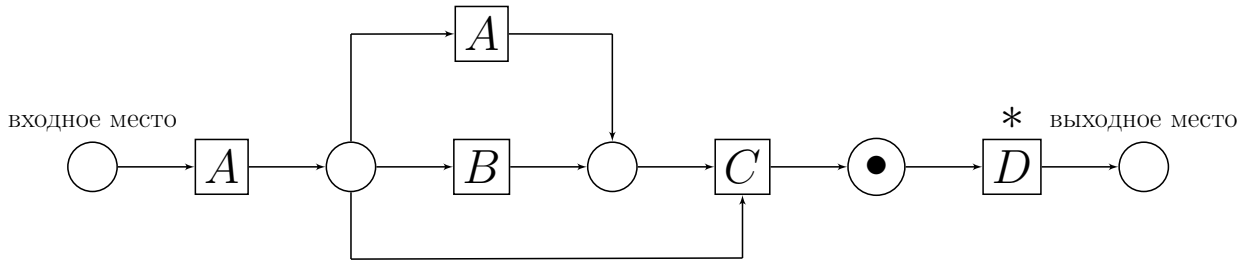

Рис. 26 — Часть сети $\mathbb{N}_3$ после срабатывания t_3

– $t_4 = D$, токен перемещается в выходное место, сеть завершает работу, Рис. 27.

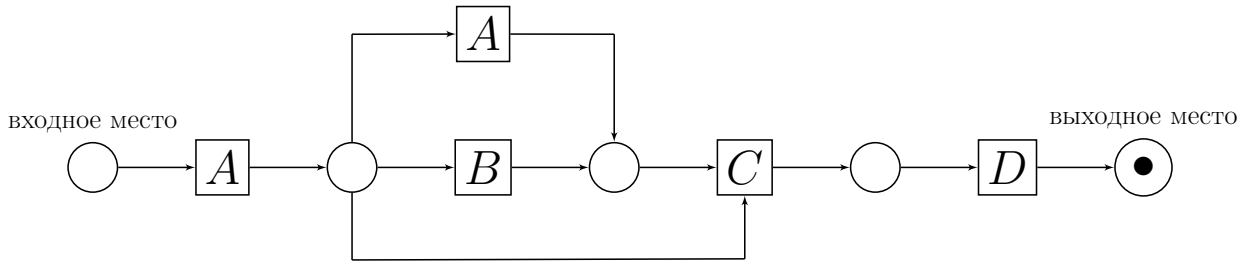

Рис. 27 — Часть сети $\mathbb{N}_3$ после срабатывания t_4

На Рис. 23-27 показаны все достижимые разметки из разметки $[i]$, каждая из которых активирует какой-либо переход сети. В сети $\mathbb{N}_3$ нет мертвых переходов.

Выполнены все свойства определения надежности, поэтому сеть $\mathbb{N}_3$ надежна. ■

Лемма 2.3. *L — лог процесса для надежной сети $\mathbb{N}_3$.*

Доказательство. По определению лога рабочего процесса для надежной сети, любое слово из лога L приводит токен из входного места сети Петри

$\mathbb{N}_3$ в выходное место. Утверждение, что L — лог рабочего процесса для сети $\mathbb{N}_3$, верно по построению. Его справедливость можно проверить перебором каждого слова в логе и просмотром передвижения токена в сети $\mathbb{N}_3$ аналогично тому, как это делалось в доказательстве надежности. ■

Лемма 2.4. *L — полный лог процесса для сети $\mathbb{N}_3$.*

Доказательство. L полный лог процесса для сети $\mathbb{N}_3$, так как L является логом процесса для сети $\mathbb{N}_3$ по Лемме 2.3, а также потому что содержит все срабатывающие последовательности для этой сети. Различие в срабатывающих последовательностях для сети $\mathbb{N}_3$ моделируется условием “ИЛИ”: префиксы трасс AAC, ABC или AC. Количество различных трасс равно 3 и эти трассы в точности совпадают с логом L. ■

Для сети $\mathbb{N}_3$ и лога L не справедливы условия Теоремы 2.1 из-за наличия повторяющегося действия A.

Построенная сеть $\mathbb{N}_3$ (Рис. 22) не является SWF-сетью, так как содержит обе запрещенные конструкции из Рис. 7.

1. Нарушение условия $\forall(p,t) \in F$, *если* $|p\bullet| > 1$, *то* $|\bullet t| = 1$, Рис. 28.

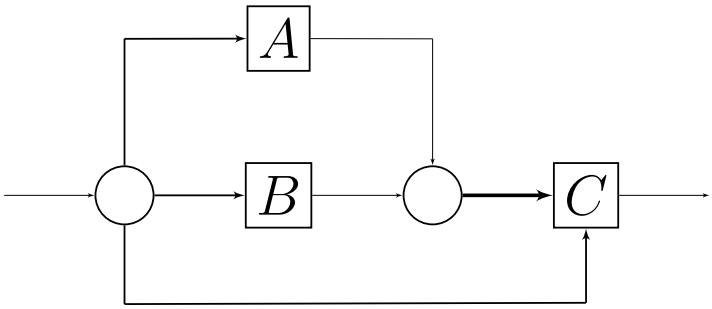

Рис. 28 — Левая запрещенная конструкция в сети $\mathbb{N}_3$

2. Нарушение условия $\forall(p,t) \in F$, *если* $|\bullet t| > 1$, *то* $|\bullet p| = 1$, Рис. 29.

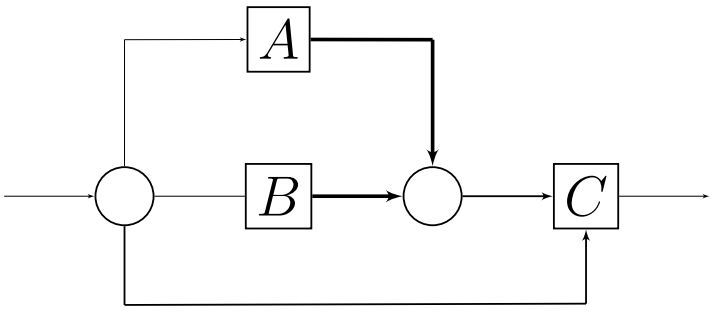

Рис. 29 — Правая запрещенная конструкция в сети $\mathbb{N}_3$

Сеть Петри $\mathbb{N}_4$ для лога L

Трудности в сети $\mathbb{N}_3$ возникают из-за того, что происходит попытка моделирования условия “ИЛИ”, где одна из возможных веток не содержит ни одного действия, то есть выбор и синхронизация должны происходить подряд. Если изменить конструкцию пустого перехода в условии “ИЛИ”, получится сеть Петри $\mathbb{N}_4$, представленная на Рис. 30. Таким образом, в сети $\mathbb{N}_4$ нет запрещенных конструкций из определения 2.16, поэтому это SWF-сеть.

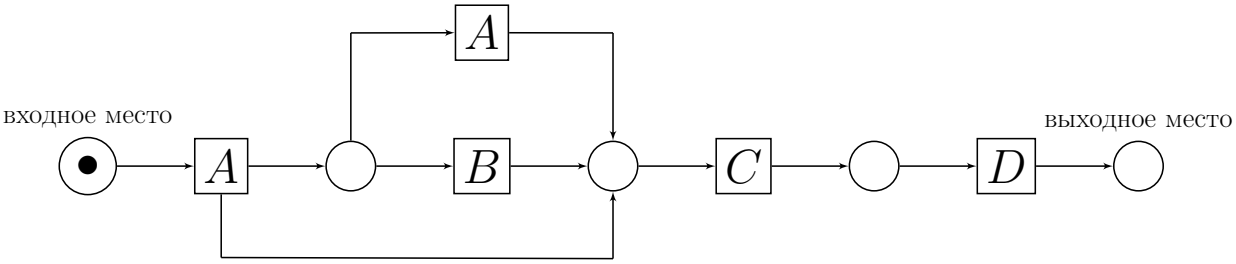

Рис. 30 — Сеть Петри $\mathbb{N}_4$ для лога L

Анализ сети $\mathbb{N}_4$. Сеть содержит 2 типа переходов:

1. входит одно ребро, исходит одно ребро (например, переходы B, C, D);
2. входит одно ребро, исходит 2 ребра (первый переход с действием A).

Так как в сети есть переход, который увеличивает количество токенов (2 тип) и нет перехода, который симметрично их уменьшает, сеть $\mathbb{N}_4$

не обладает свойством корректного завершения и поэтому не является надежной.

Лемма 2.5. *Сеть* $\mathbb{N}_4$ *не является корректно завершающейся.*

Доказательство. Свойство корректного завершения определения надежности 2.10: $\forall s \in [\mathbb{N}_4, [i]\rangle,\ o \in s \implies s = [o]$.

Для иллюстрации противоречия рассматривается срабатывающая последовательность переходов $\sigma = ACD$, переводящая метку из входного места в выходное место — $(\mathbb{N}_4, [i])[\sigma\rangle(\mathbb{N}_4, [o])$.

– Разметка $[i]$ активирует переход A, Рис. 31.

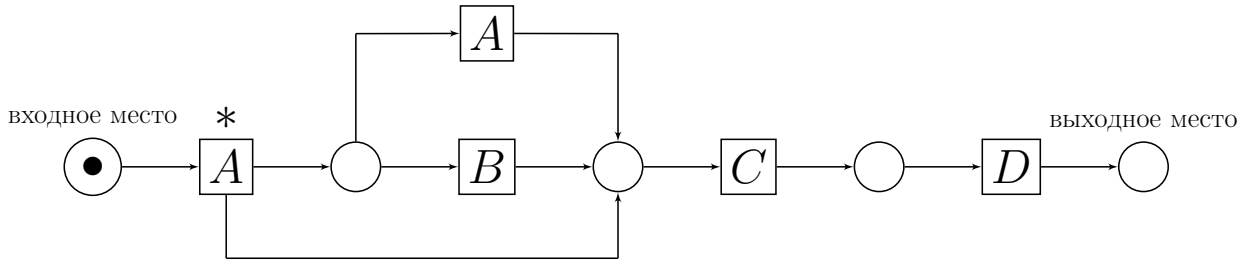

Рис. 31 — Часть сети $\mathbb{N}_4$ в изначальной разметке, активированный переход помечен $*$

– $t_1 = A$, появляется 2 метки, которые активируют 3 перехода, Рис. 32.

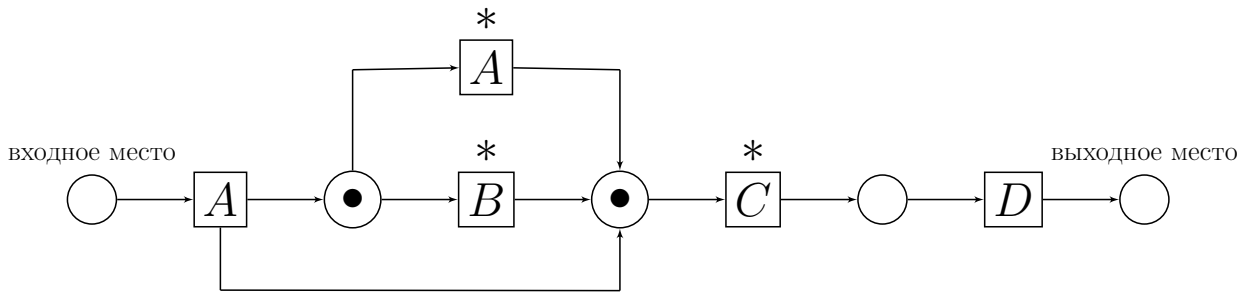

Рис. 32 — Часть сети $\mathbb{N}_4$ после срабатывания t_1

– $t_2 = C$, меняет местоположение 1 токен из двух, активированы 3 перехода, Рис. 33.
– $t_3 = D$, токен перемещается в выходное место, в сети еще остается один токен между переходами A, A и B, Рис. 34.

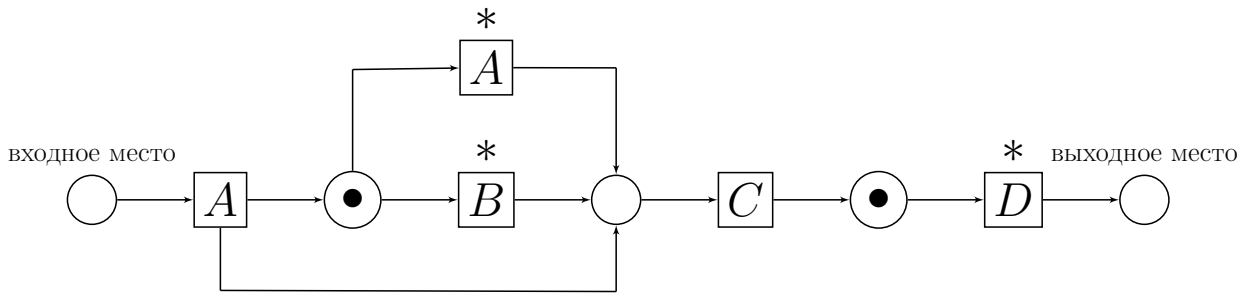

Рис. 33 — Часть сети $\mathbb{N}_4$ после срабатывания t_2

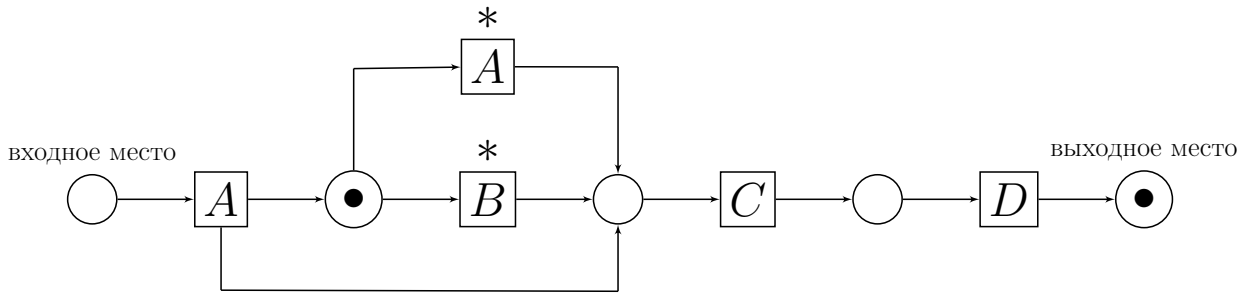

Рис. 34 — Часть сети $\mathbb{N}_4$ после срабатывания t_3

Таким образом, обнаружена достижимая разметка из разметки $[i]$ (когда в сети существовал только одна метка и она находилась во входном месте) такая, что в этой разметке есть метка, находящаяся в выходном месте и она не единственная. Сеть $\mathbb{N}_4$ не обладает свойством корректного завершения. ■

Свойство полноты лога определено для надежных сетей Петри, поэтому, так как сеть $\mathbb{N}_4$ не является надежной, то и лог L не может быть полным для этой сети.

Для сети $\mathbb{N}_4$ и лога L не справедливы условия Теоремы 2.1 из-за наличия повторяющегося действия A.

Сеть Петри $\mathbb{N}_5$ для лога L

Увеличим количество переходов в сети для того, чтобы получилась каноническая конструкция условия “ИЛИ”, как на Рис. 19. Сеть $\mathbb{N}_5$ представлена на Рис. 35.

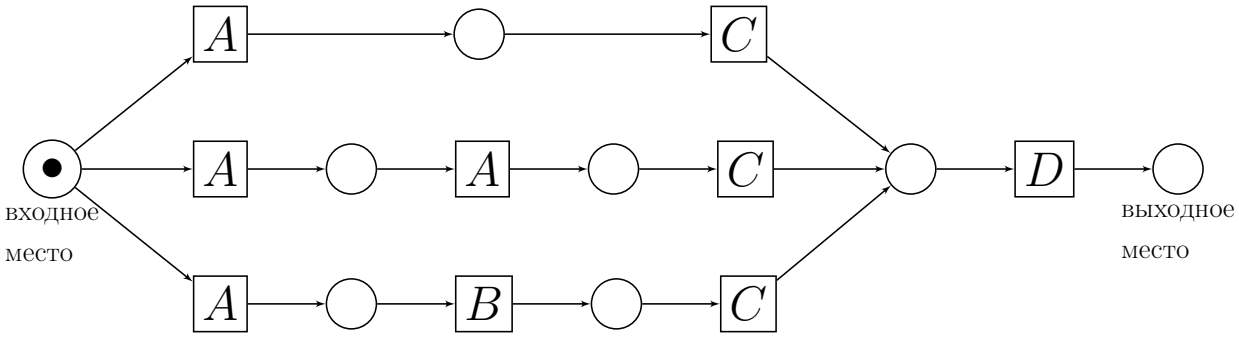

Рис. 35 — Сеть Петри $\mathbb{N}_5$ для лога L

Анализ сети $\mathbb{N}_5$. Сеть содержит 3 типа мест:

1. одно входящее ребро, одно исходящее ребро (например, место между переходами B, C);
2. несколько входящих ребер, одно исходящее ребро (место между переходами C и D);
3. одно входящее ребро, несколько исходящих ребер (входное место, при предположении одного входящего ребра).

Сеть содержит 1 тип переходов: входит одно ребро, исходит одно ребро. Поэтому сеть $\mathbb{N}_5$ будет SWF-сетью.

Лемма 2.6. *Сеть $\mathbb{N}_5$ надежна.*

Доказательство. Проверка свойств по определению надежности.

– Проверка безопасности: в каждый момент времени в каждом месте должны находиться 0 или 1 метка.

  Пусть в изначальной разметке во входном месте находится одна метка (токен) (как на Рис. 35). Наличие различных типов мест в сети не может повлиять на количество токенов, так как увеличивать количество токенов могут только переходы с несколькими

исходящими ребрами. В сети $\mathbb{N}_5$ есть 1 тип переходов, он не изменяет количества токенов, так как содержит 1 входящее ребро и 1 исходящее ребро. Поэтому сеть $\mathbb{N}_5$ безопасна.

– Проверка корректного завершения: $\forall s \in [\mathbb{N}_5, [i]\rangle$, $o \in s \implies s = [o]$, где $[i]$ — разметка, в которой есть один токен и он находится во входном месте, а $[o]$ — разметка, в которой есть один токен и он находится в выходном месте.

  Так как сеть $\mathbb{N}_5$ безопасна и если в сеть Петри не может поступать новый токен (поступление токена во входное место сети), пока текущий токен не дошел до выходного места, и потому что в сети есть только тип переходов, где переход может принять 1 токен и породить 1 токен, сеть $\mathbb{N}_5$ завершается корректно.

– Проверка отсутствия мертвых переходов.

  По определению: переход $t \in T$ *мертвый*, если не существует достижимой разметки $s' \in [\mathbb{N}_5, s\rangle$ такой, что $(\mathbb{N}_5, s')[t\rangle$.

  Рассматриваются срабатывающие последовательности переходов вида $\sigma = \delta D$, где δ одна из последовательностей AC, AAC, ABC.

  – Разметка $[i]$ активирует 3 перехода, Рис. 36.

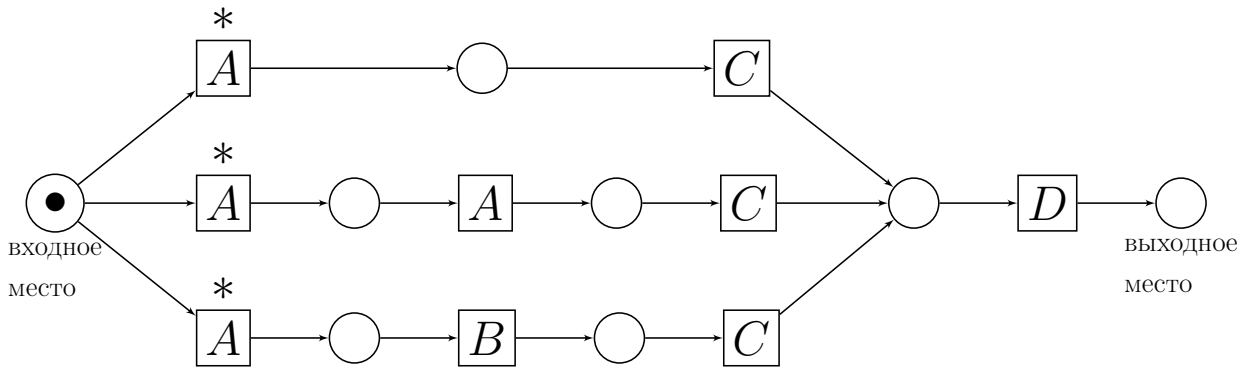

Рис. 36 — Часть сети $\mathbb{N}_5$ в изначальной разметке, активированные переходы помечены $*$

  – Моделирование условия “ИЛИ”. Возможны 3 ситуации:

    1. срабатывающая подпоследовательность $\delta = AC$, Рис. 37, Рис.38;
    2. срабатывающая подпоследовательность $\delta = AAC$, Рис. 39, Рис. 40, Рис. 38;

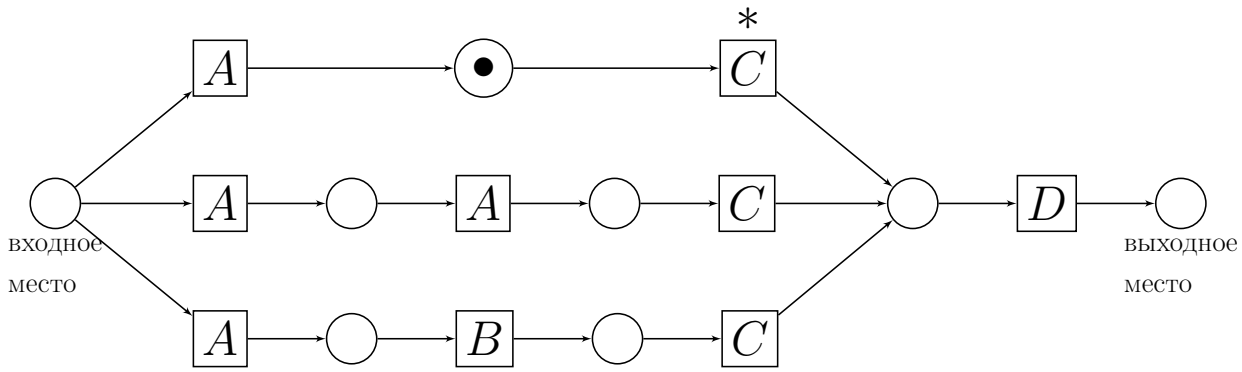

Рис. 37 — Часть сети $\mathbb{N}_5$ при срабатывании подпоследовательности $\delta = AC$, после срабатывания перехода A. Остается активирован 1 переход

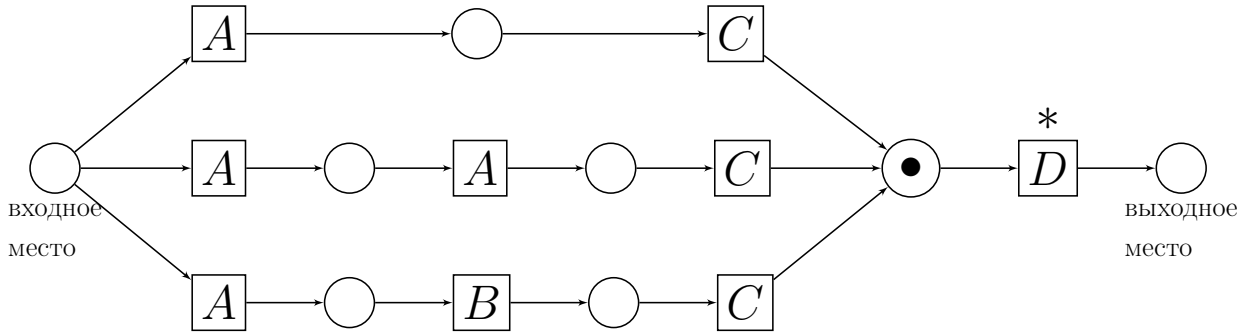

Рис. 38 — Часть сети $\mathbb{N}_5$, выход из условия “ИЛИ”

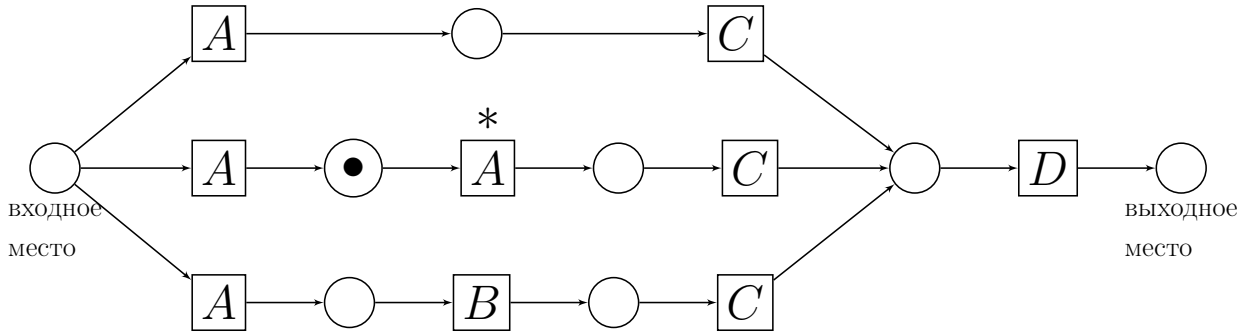

Рис. 39 — Часть сети $\mathbb{N}_5$ при срабатывании подпоследовательности $\delta = AAC$, после срабатывания первого перехода A. Остается активирован 1 переход из 3

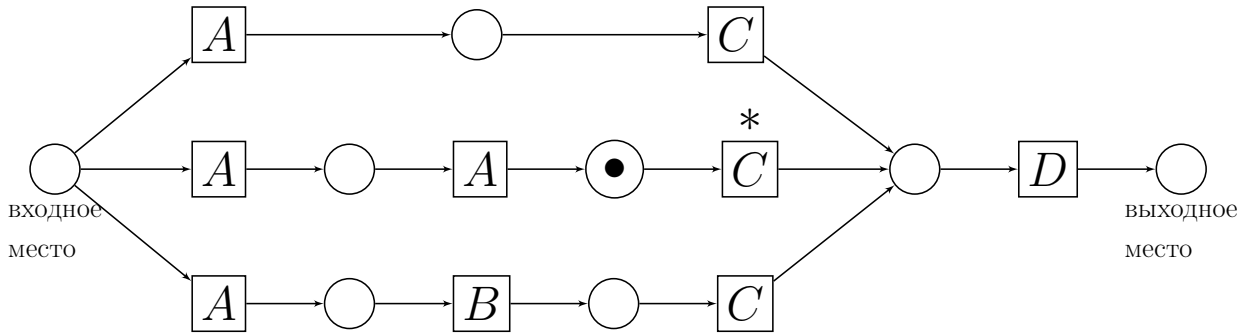

Рис. 40 — Часть сети $\mathbb{N}_5$ при срабатывании подпоследовательности $\delta = AAC$, после срабатывания второго перехода A

3. срабатывающая подпоследовательность $\delta = ABC$, Рис. 41, Рис. 42, Рис. 38.

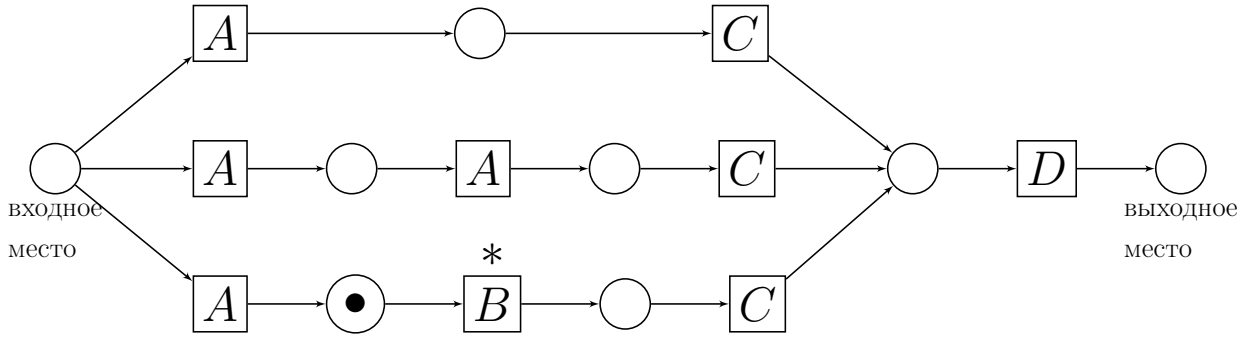

Рис. 41 — Часть сети $\mathbb{N}_5$ при срабатывании подпоследовательности $\delta = ABC$, после срабатывания перехода A. Остается активирован 1 переход из 3

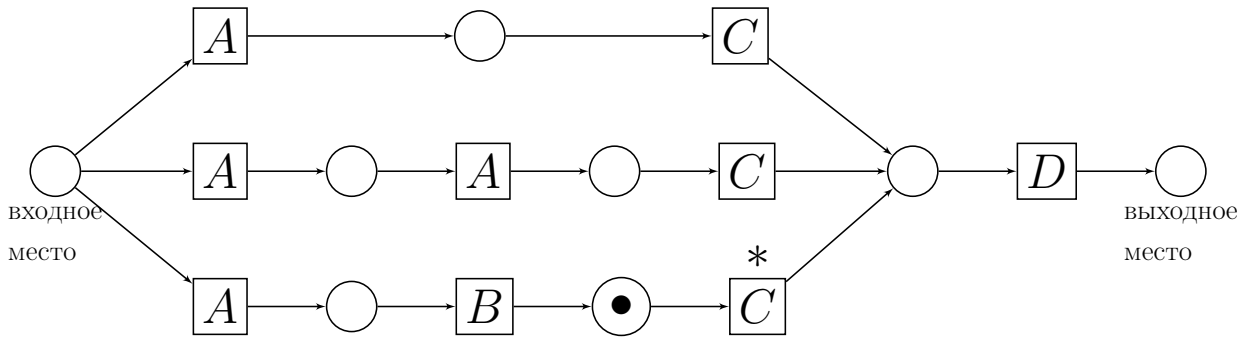

Рис. 42 — Часть сети $\mathbb{N}_5$ при срабатывании подпоследовательности $\delta = ABC$, после срабатывания перехода B

– Срабатывание перехода D, сеть завершает работу, Рис. 43.

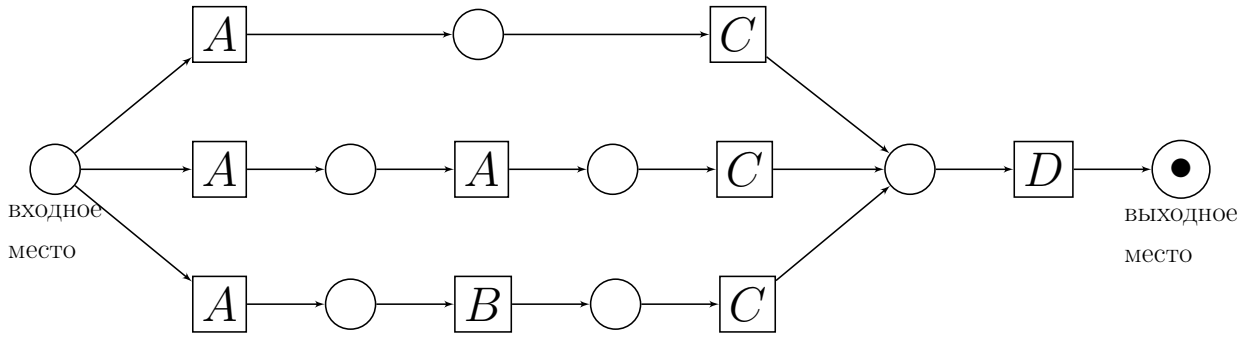

Рис. 43 — Часть сети $\mathbb{N}_5$ после срабатывания C

На Рис. 36-43 показаны все достижимые разметки из разметки $[i]$, каждая из которых активирует хотя бы один переход. Поэтому в сети $\mathbb{N}_5$ нет мертвых переходов.

Выполнены все свойства определения надежности, поэтому сеть $\mathbb{N}_5$ надежна. ■

Лемма 2.7. *L — лог процесса для надежной сети $\mathbb{N}_5$.*

Доказательство. Любое слово из лога L приводит токен из входного места в выходное. Утверждение, что L — лог рабочего процесса для сети $\mathbb{N}_5$, верно по построению. Его справедливость можно проверить перебором каждого слова в логе и просмотром передвижения токена в сети $\mathbb{N}_5$ аналогично тому, как это делалось в доказательстве надежности. ■

Лемма 2.8. *L — полный лог процесса для сети $\mathbb{N}_5$.*

Доказательство. L полный лог процесса для сети $\mathbb{N}_5$, так как является логом процесса для надежной сети по Лемме 2.7, а также потому что содержит все срабатывающие последовательности для этой сети. Различия в срабатывающих последовательностях для сети $\mathbb{N}_5$ моделируются условием “ИЛИ”. Количество различных трасс, которые может породить сеть $\mathbb{N}_5$, равно 3, и эти трассы в точности совпадают с логом L. ■

Лемма 2.9. *$\mathbb{N}_5$ — SWF-сеть.*

Доказательство. По определению: *Сеть Петри* $N = (P, T, F)$ — *SWF-сеть, если* $\forall p \in P, \forall t \in T$, *где* $(p,t) \in F$ *выполнено:*

1. *если* $|p \bullet| > 1$, *то* $|\bullet t| = 1$;
2. *если* $|\bullet t| > 1$, *то* $|\bullet p| = 1$;
3. *в сети нет неявных мест.*

Исходя из Рис. 35, в сети Петри $\mathbb{N}_5$:

1. есть одно место, из которых исходит несколько ребер. Для всех переходов, которые идут после этого места, входящее ребро единственное;
2. нет переходов, в которые входят больше, чем 1 ребро;
3. нет неявных мест по построению.

■

Сеть $\mathbb{N}_5$ не удовлетворяет условиям Теоремы 2.1 из-за наличия повторов действий. В частности, не у всех пар переходов с действием A и с действием C есть общее место. А для лога L справедливо каузальное отношение $A \rightarrow_L C$.

2.4 Построение корректной модели процесса для поиска аномалий

В следующей Теореме приведен результат поиска модели, удовлетворяющей условиям корректности для лога $L = \{ACD, AACD, ABCD\}$, описывающего процесс из 4 действий с повтором одного из действий и наличием конструкции “ИЛИ”.

Теорема 2.2. *Результаты главы 2 могут быть представлены с помощью следующей таблицы:*

Сеть Петри	*Надежность*	*SWF-сеть*	*Полнота лога*	*Т. о кауз.*
$\mathbb{N}_1$	+	—	—	+
$\mathbb{N}_3$	+	—	+	—
$\mathbb{N}_4$	—	+	—	—
$\mathbb{N}_5$	+	+	+	—

Доказательство. Доказательство надежности сети $\mathbb{N}_1$ (Рис. 10) приведено в Лемме 2.1. Сеть $\mathbb{N}_1$ не является SWF-сетью, так как содержит обе запрещенные конструкции из Рис. 7, как показано на Рис. 15 и Рис. 16. Так как сеть моделирует большее количество трасс, чем содержится в логе L, например, трассы $AAA...AACD$, $AAB...ABCD$, $ABB...BBCD$, то не выполняется свойство о полноте лога. Сеть $\mathbb{N}_1$ соответствует Теореме 2.1, так как для каждой пары действий, между которыми есть каузальное отношение, в сети есть место между соответствующими переходами.

Доказательство надежности сети $\mathbb{N}_3$ (Рис. 22) приведено в Лемме 2.2. Сеть $\mathbb{N}_3$ не является SWF-сетью, так как содержит обе запрещенные конструкции из Рис. 7, как показано на Рис. 28 и Рис. 29. Доказательство полноты лога L для сети $\mathbb{N}_3$ приведено в Лемме 2.4. Для сети $\mathbb{N}_3$ и лога L не справедливы условия Теоремы 2.1 из-за наличия повторяющегося действия A.

Из Леммы 2.5 следует, что в сети $\mathbb{N}_4$ (Рис. 30) есть переход, который увеличивает количество токенов и нет перехода, который симметрично их уменьшает, поэтому эта сеть не обладает свойством корректного завершения и поэтому не может являться надежной по определению. Так как сеть $\mathbb{N}_4$ была получена модификацией сети $\mathbb{N}_3$, чтобы в ней не было запрещенных конструкция из определения 2.16, то эта сеть является SWF-сетью по построению. Свойство полноты лога определено для надежных сетей Петри, поэтому, так как сеть $\mathbb{N}_4$ не является надежной, то и лог L не может быть полным для этой сети. Для сети $\mathbb{N}_4$ и лога L не справедливы условия Теоремы 2.1 из-за наличия повторяющегося действия A.

Доказательство надежности сети $\mathbb{N}_5$ (Рис. 35) приведено в Лемме 2.6. Доказательство того, что сеть $\mathbb{N}_5$ является SWF-сетью, приведено в Лемме 2.9. Доказательство полноты лога L для сети $\mathbb{N}_5$ приведено в Лемме 2.8. Сеть $\mathbb{N}_5$ не удовлетворяет условиям Теоремы 2.1 из-за наличия повторов действий. В частности, не у всех пар переходов с действием A и с действием

C есть общее место. А для лога L справедливо каузальное отношение $A \rightarrow_L C$.

■

В таблице Теоремы 2.2 не приведена сеть Петри $\mathbb{N}_2$, так как данная сеть показывает, что переходы *OR-split*, *OR-join* без наложения дополнительных условий на рассматриваемый класс сетей Петри не работают корректно. Более точно, эти переходы работают в точности как переходы *AND-split*, *AND-join* и моделируют другой процесс.

Показано, что Теорема о каузальности не выполняется, когда в сети Петри есть переходы с одинаковыми действиями. Также на примере сети $\mathbb{N}_2$ показано, что Теорема о каузальности не выполняется и при наличии блоков *OR-split*, *OR-join*, *AND-split*, *AND-join*.

Теорема 2.2 с точки зрения нахождения подхода к решению задачи поиска аномалий свидетельствует о том, что даже для простого процесса наложение нескольких естественных (необходимых для дальнейшего поиска аномалий) условий корректности приводит к высокой сложности поиска подходящей модели. Решение задачи по сути является переборным [17] и не гарантирует успешного результата.

2.5 Выводы

В настоящей главе представлен подход к решению задачи поиска аномалий с использованием модели в терминах сетей Петри, которая была восстановлена по логу процесса. Проведен анализ совместимости следующих свойств.

1. Свойство надежности.
2. Свойство запрета последовательного выполнения в сети конструкций синхронизации и выбора, определение структурированной сети Петри.
3. Полнота лога рабочего процесса для рассматриваемой сети.

4. Сохранение в сети отношения каузальности между переходами, если между соответствующими действиями в логе это отношение было выполнено, Теорема о каузальности.

Приведен пример простого процесса, состоящего из 4 действий. В процессе есть повтор действий A и конструкция "ИЛИ", допускающая переход без выполнения какой-либо действия (трасса AC). Для этого процесса рассмотрены 5 естественных моделей, представленных в виде сети Петри. Доказан ряд утверждений относительно рассматриваемых свойств корректности. Показано, почему каждую из моделей нельзя назвать подходящей под базовые требования, которые обычно налагаются на модели в терминах сетей Петри.

Для эффективного решения задачи поиска аномалий необходимым условием является наличие модели, которая бы описывала поведение процесса. Более того, необходимо, чтобы данная модель восстанавливалась по полному логу единственным образом и при этом не порождала новых трасс, которых не было в исходном логе. Однако приведенный пример процесса показывает, что сложности возникают не только с построением единственно-возможной модели процесса, но и с произвольной моделью, которая бы удовлетворяла базовым свойствам корректности, которые обычно налагаются на модель в терминах сетей Петри. Таким образом, продемонстрирована сложность однозначного восстановления модели в терминах сетей Петри с наложением нескольких условий корректности. Этот факт затрудняет дальнейшее решение задачи поиска аномалий с использованием такой модели процесса.

Показано, что для простого примера невозможно построить решение задачи поиска аномалий, используя для этого модель в терминах сети Петри, так как уже построение однозначной математической модели, которая удовлетворяла бы базовым свойствам корректности, является трудоемкой задачей. Этот факт не говорит о том, что для всех реальных процессов использование математической модели в виде сетей Петри для их описания является нежелательным, но для последующей формулировки и поиска решений задач обеспечения информационной безопасности важна непротиворечивость и корректность математической модели, которая описывает процесс. Тем более, подход к решению задачи поиска аномалий

часто необходимо обобщать на случай одновременного функционирования нескольких процессов. В настоящей главе показано, что проблемы построения однозначной, корректной модели, возникают уже для лога событий, описывающего простой процесс.

Глава 3
Выявление аномалий с помощью моделей, заданных ациклическими ориентированными графами

В настоящей главе рассматриваются методы и алгоритмы для решения задачи поиска аномалий в случае наличия нескольких эталонных процессов при некоторых ограничениях на структуру этих процессов. В качестве формальной модели процессов рассматривается ациклический ориентированный граф. Результаты главы представлены в работах [70; 73].

Сперва рассматривается частный случай, когда при построении модели процесса в трассе лога может содержаться исполнение только одного процесса. Для этого случая представлены результаты исследования сложностных характеристик решения задачи поиска аномалий, когда процессы состоят из различных действий и когда пересечение действий процессов является конечным множеством.

Затем рассматривается случай, когда в одной трассе содержатся исполнения нескольких процессов. Предлагается достаточное условие, налагаемое на трассы лога, такое, что наличие в трассах исполнений нескольких процессов не представляет собой усложнение рассматриваемых задач, а позволяет применить алгоритмы построения процессов и поиска аномалий аналогично случаю, когда в одной трассе лога может содержаться исполнение только одного процесса.

Получены оценки сложности построения формальных моделей для процессов в зависимости от свойств лога, также даны оценки сложности ответа на вопрос, является ли поступающая трасса аномальной относительно уже построенных моделей процессов.

Для задачи построения модели процесса предполагается, что известен некоторый журнал исполнений (лог) эталонного процесса, по которому строится модель. В качестве формальной модели могут рассматривать-

ся конечный автомат [18; 19], сеть Петри [16; 20], вероятностные модели [12; 21; 22], ориентированный граф [11; 23; 24], множество ассоциативных правил [13; 25] и другие. У каждой из моделей есть свои достоинства и недостатки. Так, например, модели в виде сетей Петри обладают большой выразительной мощностью, однако, как было показано автором в главе 2, построение корректной сети Петри для лога может быть трудоемкой задачей. В задаче поиска аномалий обычно происходит сопоставление новых исполнений с имеющейся моделью процесса. Если исполнение не может быть представлено в построенной модели, исполнений называется “аномальным”.

В этой главе рассматривается подход представления процессов в множестве ориентированных ациклических графов (Directed Acyclic Graph — DAG). Достоинством данного подхода является хорошая интерпретируемость полученной модели, линейная сложность построения модели относительно размера лога. Недостатком данного метода является ограничение, налагаемое на модель процесса, заключающееся в отсутствии ориентированных циклов. Это обстоятельство сужает класс логов, на которых метод может быть применен.

В настоящей работе рассматривается расширение трех алгоритмов, предложенных в работе [11], с моделирования одного до моделирования нескольких процессов в рамках одного лога исполнений, оценивается сложность предложенных алгоритмов.

3.1 Основные определения

Модель одного процесса строится в виде ориентированного графа без ориентированных циклов. Вершины графа задают действия процесса, действия предполагаются атомарными. Дуги графа задают причинно-следственные связи между действиями. Таким образом, если полагается, что действие A должно быть выполнено до действия B, то в модели должна

появиться дуга или ориентированный путь, ведущий из A в B. Среди множества всех ориентированных ациклических графов на модель процесса при заданном логе налагаются следующие условия:

- *полнота* – граф должен содержать все зависимости, имеющиеся в логе;
- *неизбыточность* – граф не должен порождать зависимостей, которых не было в логе процесса;
- *минимальность* – граф должен состоять из минимального количества дуг.

В отличие от работы [11], в определении процесса не рассматриваются функция переходов и функция выходов. Полагается, что *процесс* P задается с помощью конечного множества действий V и ориентированного графа $G = (V, E)$.

Пусть граф G имеет один *исток* – вершину, в которую не следует ни одна дуга, есть только исходящие дуги, и один *сток* – вершину, из которой не исходит дуг. Таким образом, налагается ограничение, что в рамках одного процесса все его исполнения начинаются с одного и того же действия и заканчиваются одним и тем же действием. Если в рамках процесса нет такого начального и конечного действий, то их можно добавить искусственно. В диссертации полагается, что каждая информационная технология имеет свое начальное и конечные действия, которые при этом не обязательно должны быть уникальными среди всех функционирующих одновременно информационных технологий.

Далее вводится ряд определений по аналогии с работой [11]. Некоторые из определений были модифицированы для удобства их использования в задаче поиска аномалий. Пусть задан конечный непустой алфавит V возможных действий для процесса P:

Определение лога, трассы и действия для этой главы совпадает с Определением 2.11 главы 2. Например, запись лога $L = \{ABC, DF\}$ означает, что в нем содержатся 2 трассы и множество действий $\{A, B, C, D, F\}$.

Если в процессе существует зависимость между двумя действиями, то эти действия должны появляться в трассах лога в соответствующем порядке. Однако, о зависимостях действий в процессе возможно узнать только по тем зависимостям, которые представлены в логе процесса, поэто-

му определение зависимости вводится для лога. В модели процесса каждая зависимость представляется либо как направленная дуга, либо как направленный путь от одного действия к другому.

Аналогично главе 2, далее водятся определения частичного предпорядка (отношение предшествия в Определении 2.12) и зависимости действий (прямое каузальное отношение в Определении 2.12). В этой главе будет полагаться, если между некоторой парой действий выполнено отношение потенциального параллелизма или они не встречаются вместе согласно Определению 2.12, то эта пара действий независимы.

Определение 3.1 (Частичный предпорядок для лога)**.** Пусть задан лог L одного процесса P. Действие B *следует* за действием A ($A \lesssim_L B$), если в логе L существует некоторая трасса такая, что в этой трассе действие B начинается после того, как заканчивается выполнение действия A, либо существует трасса в логе L с действиями A, B, C такая, что C следует за A, а B следует за C.

Например, для лога $L = \{ABA, DF\}$ справедливо: $A \lesssim_L B$, $B \lesssim_L A$, $A \lesssim_L A$, $D \lesssim_L F$.

Определение 3.2 (Зависимость действий)**.** Пусть задан лог L одного процесса P. Действие B *зависит* от действия A ($A \to_L B$), если B следует за A, а A не следует за B. Если A следует за B и B следует за A, либо A не следует за B и B не следует за A, то действия A и B *независимы*.

Так, для лога $L = \{ABA, DF\}$ справедливо: $D \to_L F$.

Определения частичного предпорядка и зависимости действий могут быть заданы и для одной трассы. Далее, если не сказано обратного, записи $A \lesssim B$, $A \to B$ обозначают частичный предпорядок и зависимость действий в логе L.

Определение 3.3 (Граф зависимостей)**.** Пусть задано множество действий V и лог L некоторого процесса. Ориентированный граф G называется *графом зависимостей*, если ориентированный путь из A в B в графе G существует тогда и только тогда, когда B зависит от A.

Для заданного лога может существовать несколько графов зависимостей. Два графа с одинаковым транзитивным замыканием множества ребер представляют собой одинаковое множество зависимостей [66].

Можно ввести определение *подграфа G', порождаемого трассой* ω, в графе зависимостей $G = (V,E) : G' = (V',E')$, где $V' = \{A \in V | A \in \omega\}$ и $E' = \{(B, A) \in E |\ B \lesssim_\omega A\}$, где $\lesssim_\omega$ — отношение частичного порядка на действиях, задаваемое трассой ω.

Определение 3.4 (Согласованность трассы с графом зависимостей)**.** Пусть задан граф зависимостей процесса P $G = (V,E)$ и трасса ω. Трасса ω согласуется с G, если подграф $G' = (V',E')$, порождаемый трассой ω, связен; первое и последнее действия трассы ω — действия, начинающее и завершающее процесс P; все вершины в V' достижимы из начального действия; никакая зависимость в графе G не нарушена упорядочиванием действий в ω.

Не все графы зависимостей, содержащие в себе зависимости некоторого лога L, являются корректными. Для формализации ограничений, которые обычно налагаются на графы зависимостей, чтобы графы зависимостей могли называться моделями процесса, вводится определение *графа, конформного логу*:

Определение 3.5 (Конформный логу граф)**.** Граф зависимостей G называется *конформным* логу L, если выполнены следующие условия:

- для каждой зависимости в L существует ориентированный путь в G;
- между независимыми действиями в L не существует ориентированного пути в G;
- каждая трасса лога L согласована с G.

Определение 3.6. Пусть задана трасса $\omega = A_1 \dots A_q$. *Подстрокой трассы* ω *длины* $r \leqslant q$ называется трасса $\omega' = A_i A_{i+1} \dots A_{i+r-1}$, где $i \in [1, q - r + 1]$.

3.2 Постановка задачи поиска аномалий

В работе [11] были предложены три алгоритма и рассчитана временная сложность построения модели в виде ациклического ориентированного графа, в зависимости от различных свойств лога исполнений одного процесса. Пусть $m = |L|$ — количество трасс в логе, $n = |V|$ — количество возможных действий процесса P. Предполагается, что $m \to \infty$, $n \to \infty$. Основным предположением о входных параметрах алгоритмов является предположение, что количество трасс m в логе L превышает количество действий n, $m >> n$.

Алгоритмы, которые строят модель процесса в виде графа по заданному логу L, предложенные в работе [11], в общем виде можно представить в виде следующих шагов:

1. добавление в граф всех дуг, соответствующих зависимостям между действиями, представленными в логе L;
2. удаление дуг внутри сильно связных компонент графа;
3. добавление меток на дуги, которые присутствуют в транзитивных замыканиях подграфов, порождаемых трассами лога L. Удаление дуг без меток;
4. склейка вершин, соответствующих одинаковым действиям.

Оценки сложности алгоритмов из работы [11] используются в настоящей диссертации в виде лемм. Под элементарными операциями для вычисления оценок сложности алгоритмов понимаются простейшие операции редактирования над графами (например: добавление/удаление вершины, добавление/удаление ребра, слияние/расщепление вершин).

Алгоритм 3.1 (Построение конформного DAG специального вида [11])**.** *Пусть задан лог процесса L. Известно, что L содержит только трассы, в которых каждое действие алфавита V встречается ровно по одному разу. Известно, что конформный логу L граф $G = (V, E)$ не содержит циклов. Конформный граф G может быть построен следующим образом.*

1. *$E = \varnothing$, V – множество всех действий в L (может быть инициализировано на следующем шаге алгоритма).*

2. *Для каждой трассы* $\omega \in L$ *и для каждой пары действий* $u, v \in \omega$ *таких, что* u *встречается раньше, чем* v *– добавить дугу* (u, v) *в множество* E.
3. *Удалить из* E *дуги, которые встречаются в двух направлениях.*
4. *Вычислить транзитивное замыкание графа* G.
5. *Вернуть граф* $G = (V, E)$.

Лемма 3.1 (О сложности построения конформного DAG специального вида [11]). *Пусть задан лог процесса* L. *Известно, что* L *содержит только трассы, в которых каждое действие алфавита* V *встречается ровно по одному разу. Известно, что конформный логу* L *граф* G *не содержит циклов. Сложность алгоритма построения конформного графа* G *составляет* $O(mn^2)$.

Алгоритм 3.2 (Построение конформного DAG [11]). *Пусть задан лог процесса* L. *Известно, что конформный логу* L *граф* $G = (V, E)$ *не содержит циклов. Конформный граф* G *может быть построен следующим образом.*

1. $E = \varnothing$, V *– множество всех действий в* L *(может быть инициализировано на следующем шаге алгоритма).*
2. *Для каждой трассы* $\omega \in L$ *и для каждой пары действий* $u, v \in \omega$ *таких, что* u *встречается раньше, чем* v *– добавить дугу* (u, v) *в множество* E.
3. *Поиск сильно-связных компонент в графе* G. *Удаление дуг из* E, *которые находятся в одной сильно-связной компоненте.*
4. *Для каждой трассы* $\omega \in L$:
    а) *поиск подграфа в* G, *порождаемого трассой* ω;
    б) *вычисление транзитивного замыкания подграфа;*
    в) *пометка дуг в* E, *которые содержатся в транзитивном замыкании.*
5. *Удаление неотмеченных дуг из* E.
6. *Вернуть граф* $G = (V, E)$.

Лемма 3.2 (О сложности построения конформного DAG [11]). *Пусть задан лог процесса* L. *Известно, что конформный логу* L *граф* G *не содержит циклов. Сложность алгоритма построения конформного графа* G *составляет* $O\left(mn^3\right)$.

Алгоритм 3.3 (Построение конформного графа [11]). *Пусть задан лог процесса L. Конформный логу L граф $G = (V, E)$ может быть построен следующим образом.*

1. *$E = \varnothing$, V – множество всех действий в L (может быть инициализировано на следующем шаге алгоритма).*
2. *Для каждой трассы $\omega \in L$ перенумеровать одинаковые действия, тем самым перейти к новому увеличенному множеству вершин V' и графу $G' = (V', E')$.*
3. *Для каждой трассы $\omega \in L$ и для каждой пары действий $u, v \in \omega$ таких, что u встречается раньше, чем v – добавить дугу (u, v) в множество E'. (Шаги 1-3 алгоритма возможно осуществить за один проход по логу).*
4. *Поиск сильно-связных компонент в графе G'. Удаление дуг из E', которые находятся в одной сильно-связной компоненте.*
5. *Для каждой трассы $\omega \in L$:*
   - *а) поиск подграфа в G', порождаемого трассой ω;*
   - *б) вычисление транзитивного замыкания подграфа;*
   - *в) пометка дуг в E', которые содержатся в транзитивном замыкании.*
6. *Удаление неотмеченных дуг из E'.*
7. *Объединить вершины графа G', которые соответсвуют одинаковым действиям графа G.*
8. *Вернуть граф $G = (V, E)$.*

Лемма 3.3 (О сложности построения конформного графа [11]). *Пусть задан лог процесса L. Сложность алгоритма построения конформного логу L графа G составляет $O\left(m\,(kn)^3\right)$, где k – максимальное количество повторов некоторого действия в логе.*

Первая задача, решение которой представлено в главе: пусть задан некоторый лог L, который может содержать в себе трассы (исполнения) нескольких процессов, необходимо построить конформные графы (модели) этих процессов и оценить трудоемкость построения.

Вторая задача, решение которой представлено в главе: пусть задана трасса ω, $u = |\omega| \to \infty$, которая может содержать в себе подтрассы (исполнения) нескольких процессов, необходимо, используя построенные модели процессов, ответить на вопрос является ли данная трасса корректным исполнением некоторого процесса (процессов).

3.3 Поиск аномалий с использованием модели процесса, построенной по логу, который содержит данные нескольких процессов

Рассматриваются s процессов $P_1, \ldots, P_s$, с непустыми множествами действий $V_1, \ldots, V_s$ и графами $G_1 = (V_1, E_1), \ldots, G_s = (V_s, E_s)$ соответственно.

Пусть задан некоторый лог L, $M = |L| \to \infty$, $n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$, k – максимальное количество повторов действия в логе L. Также, как и в работе [11], предполагается, что $M >> n$, количество процессов s процессов конечно.

Задача, решение которой представлено в главе: пусть задан некоторый лог L, который может содержать в себе трассы (исполнения) нескольких процессов, необходимо построить конформные графы (модели) этих процессов и оценить трудоемкость данного построения.

Рассматривается случай, когда s процессов имеют различные множества действий. Простейшим примером такого лога при $s = 2$ является лог $L = \{AB, DF\}$.

Теорема 3.1 (Процессы с различными множествами действий)**.** *Пусть задан лог L для s процессов. Пусть выполнены условия Лемм 3.1, 3.2, 3.3. Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \varnothing$, при $i \neq j, i, j = 1, \ldots, s$. Тогда сложность построения*

s конформных графов составляет $O\left(Mn^2\right), O\left(Mn^3\right), O\left(M\left(kn\right)^3\right)$ *соответственно, в зависимости от свойств лога* L.

Доказательство. Так как множества действий для каждого из процессов не пересекаются, Алгоритмы 3.1, 3.2, 3.3 не требуют модификации и работают корректно и на s процессах. Связано это со структурой алгоритмов, которая состоит в том, что на начальных шагах в граф зависимостей добавляются дуги, задающие все зависимости лога. Далее происходит редуцирование дуг получившегося графа. Тем самым, в графе зависимостей остается минимально возможное количество дуг с сохранением необходимых зависимостей лога.

По условию теоремы, множества действий в процессах не пересекаются. Таким образом, дуги между процессами не могут быть добавлены на первом шаге алгоритмов, поэтому Алгоритмы 3.1, 3.2, 3.3 корректно сначала построят, а потом минимизируют s различных компонент связности, каждая из которых задает свой граф зависимостей (модель) процесса. Условия Лемм 3.1, 3.2, 3.3 на лог L выполняются, таким образом, сложность алгоритмов остается такой же, как в Леммах 3.1, 3.2, 3.3 с поправкой на увеличившийся размер лога M. ■

Если построено s конформных графов зависимостей для s процессов и известно, что в поступающей трассе ω может быть исполнение только одного процесса, то задача поиска аномалий в ω сводится к задаче определения согласованности этой трассы с s графами зависимостей. Таким образом, алгоритм для решения задачи поиска аномалий в трассе будет выглядеть следующим образом:

Алгоритм 3.4 (Поиск аномалий в трассе при наличии множества конформных графов)**.** *Пусть построено* s *конформных графов. Известно, что* ω *может содержать в себе исполнение одного процесса. Поиск аномалий в некоторой трассе* ω *может быть выполнен следующим образом.*

1. V' *– множество действий в* ω.
2. *Определение соответствующего трассе* ω *конформного графа* $G = (V, E)$ *из построенных конформных графов.*
3. *Проверка равенства первого и последнего действия в* ω *с стоком и истоком графа* G.

4. *Проверка* $V' \subseteq V$.
5. *Проверка достижимости вершин из* V' *из начальной вершины (истока) графа* G.
6. *Построение множества* E', *как построение отношения* $\rightarrow_{\omega}$.
7. *Проверка* $E' \subseteq E$.
8. *Проверка связности графа* G'.
9. *Трасса* ω *содержит аномалии, если какая-либо из проверок не была пройдена. Иначе – трасса* ω *не содержит аномалий.*

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$, $i = 1, \ldots, s$, $u = |\omega|$ — длина трассы ω, $u \to \infty$.

Теорема 3.2 (Поиск аномалий в процессах с различными множествами действий)**.** *Пусть* s *процессов имеют попарно различные множества действий,* $V_i \cap V_j = \varnothing$, *при* $i \neq j, i, j = 1, \ldots, s$. *Пусть для* s *процессов построено* s *конформных графов зависимостей, тогда поиск аномалий в некоторой трассе* ω *может быть выполнен за не более, чем* $s + C_1 un + C_2 en^2$ *операций, где* C_1, C_2 — *некоторые константы.*

Доказательство. Пусть $G' = (V', E')$ — граф, порождаемый трассой ω. Поиск аномалий в трассе ω возможно осуществить посредством проверки согласованности трассы с одним из построенных конформных графов зависимостей, то есть посредством применения Алгоритма 3.4.

Сложность шагов Алгоритма 3.4 при выполнении условий Теоремы 3.1 можно оценить следующим образом.

1. Инициализация V' множеством действий в ω, u операций.
2. Определение соответствующего трассе ω конформного графа: так как действия между процессами для Теоремы 3.1 не пересекаются, то достаточно s операций, чтобы сравнить, например, начальные действия построенных графов и начальное действие проверяемой трассы ω. Пусть нужный конформный граф для проверки соответствия трассы ω — граф $G = (V, E)$.
3. Проверка равенства первого и последнего действия в ω с стоком и истоком графа G. Сложность шага не превысит некоторую константу C.

4. Проверка $V' \subseteq V$. Проверку можно осуществить за $|V'| \leqslant u$ сравнений букв с $|V| \leqslant n$ вершинами, не более un операций.
5. Проверка достижимости вершин из V' из начальной вершины графа $G = (V, E)$. На этом шаге уже известно, что количество различных действий в трассе $\boldsymbol{\omega}$ не превышает $|V'| \leqslant n$. Поиск всех достижимых из начальной вершин можно осуществить с помощью одного запуска алгоритма поиска в ширину [67], сложность $O(n + e)$. После чего сравнить вершины из V' с получившимся списком достижимых вершин из начальной вершины графа G, что составит не более n^2 операций. Сложность шага не превысит $C'(n + e) + n^2$, где C' – некоторая константа.
6. Построение множества E'. Сначала происходит построение отношения $\lesssim_\omega$, а потом построение отношения $\rightarrow_\omega$.

   За один проход по трассе $\boldsymbol{\omega}$ для каждого из возможных, не более, чем n, действий возможно определить первую и последнюю позицию, на которых встречалось это действие в трассе, сложность составит u операций сравнения и не более, чем $2n$ дополнительной памяти для хранения номеров позиций.

   Пусть хранение отношения $\lesssim_\omega$ организовано с помощью хеш-таблицы, где ключи — уже встреченные в трассе $\boldsymbol{\omega}$ пары действий, значения — встречалась ли пара в обратном порядке, для хранения пар потребуется не более, чем $n^2/2$ дополнительной памяти.

   Используя информацию о первом и последнем вхождении каждого из действий, за не более, чем n^2 операций возможно построить отношение $\lesssim_\omega$. Вставка нового ключа в хеш-таблицу происходит только в том случае, если проверяемая пара действий не является парой в обратном порядке для некоторого ключа, который в хеш-таблице уже присутствует (в таком случае увеличить значение по этому ключу). Проверка наличия ключа в хеш-таблице осуществляется за $O(1)$.

   Построение отношения $\rightarrow_\omega$ возможно осуществить за не более, чем $u + n^2$ операций и не более, чем $n^2/2 + 2n$, дополнительной памяти для хранения пар действий.

7. Проверка $E' \subseteq E$. Так как на этом шаге уже известно, что $|V'| \leqslant n$, поэтому $|E'| \leqslant n^2$. Так как $|E| \leqslant e$ проверка займет не более en^2 операций. Если $E' \subseteq E$ выполнено, то можно говорить, что никакая из зависимостей графа G не нарушена упорядочиванием действий из w.
8. Проверка связности графа G'. На этом шаге уже известно, что количество ребер в E' не превышает e, поэтому проверку связности возможно осуществить с помощью одного запуска алгоритма поиска в ширину, сложность $O(n+e)$, не более $C''(n+e)$ операций, где C'' – некоторая константа.

Итоговая сложность проверки согласованности трассы по вышеперечисленным шагам составляет не более $u + s + C + un + (C'(n+e) + n^2) + (u + n^2) + en^2 + C''(n+e) \leqslant s + C_1un + C_2en^2$, где C_1, C_2 – константы. ■

Если для задачи поиска аномалий, построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 3.1 (Поиск аномалий в процессах с различными множествами действий)**.** *Пусть s процессов имеют попарно различные множества действий, $V_i \cap V_j = \varnothing$, при $i \neq j, i, j = 1, \ldots, s$. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за $O(u)$.*

Пусть $m = \max_i |L_i| \to \infty$, L_i – часть лога L, соответствующая i-му процессу, $i = 1, \ldots, s$. Пусть $n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$, k — максимальное количество повторов действия в логе L. Пусть известно, что процесс однозначно можно идентифицировать его начальным действием. Простейшим примером лога для двух процессов может быть лог $L = \{AB, CB\}$. При условии s различных начальных действий, справедлива следующая теорема.

Теорема 3.3 (Процессы, отличимые по начальному действию)**.** *Пусть задан лог L для s процессов. Пусть s процессов имеют попарно различные начальные действия. Тогда сложность построения s конформных графов*

составляет $O\left(mn^2\right)$, $O\left(mn^3\right)$, $O\left(m\left(kn\right)^3\right)$ *соответственно, в зависимости от свойств лога* L.

Доказательство. Так как процессы имеют попарно различные начальные действия, то за $M = |L|$ операций, можно разбить лог L на частичные логи L_i такие, что $L_i \cap L_j = \varnothing, i \neq j, i, j = 1, \ldots, s$. Для хранения соответствия начального действия процессу потребуется s дополнительной памяти.

Так как часть лога L_i соответствует логу одного процесса, то в зависимости от свойств этой части лога будут справедливы условия одного из Алгоритмов 3.1, 3.2, 3.3. Решение задачи построения s конформных графов может быть осуществлено за s запусков этих алгоритмов.

Так как $M \leqslant sm$, то справедливы следующие верхние оценки в зависимости от свойств лога L.

- Лог каждого из s процессов содержит только трассы, в которых каждое действие алфавита V_i встречается ровно по одному разу. Известно, что соотвествующие конформные логу L_i графы G_i не содержат циклов. Используя Лемму 3.1, s запусков Алгоритма 3.1 будет иметь оценку сложности $O\left(mn^2\right)$.
- Лог каждого из s процессов не содержит циклов. Используя Лемму 3.2, s запусков Алгоритма 3.2 будет иметь оценку сложности $O\left(mn^3\right)$.
- Пусть существует часть лога L_i, которая не удовлетворяет условиям Леммы 3.2. Используя Лемму 3.3, в худшем случае, s запусков Алгоритма 3.3 будет иметь оценку сложности $O\left(m\left(kn\right)^3\right)$, где k — максимальное количество повторов действия среди действий в логе.

■

Пусть $n = \max_i |V_i| \to \infty$, $e = \max_i |E_i|$,$i = 1, \ldots, s$, $u = |\omega| \to \infty$ — длина трассы ω.

Теорема 3.4 (Поиск аномалий в процессах, отличимых по начальному действию)**.** *Пусть* s *процессов имеют попарно различные начальные действия. Пусть для* s *процессов построено* s *конформных графов зависимостей, тогда поиск аномалий в некоторой трассе* ω *может быть*

выполнен за не более, чем $s + C_1un + C_2en^2$ *операций, где* C_1, C_2 *— некоторые константы.*

Доказательство. Поиск аномалий в трассе ω возможно осуществить посредством проверки согласованности трассы с одним из построенных конформных графов зависимостей, то есть посредством применения Алгоритма 3.4.

Сложность шагов Алгоритма 3.4 при выполнении условий Теоремы 3.3 можно оценить следующим образом.

1. Инициализация V' множеством действий в ω, u операций.
2. Определение соответствующего трассе ω конформного графа: так как по условию s процессов имеют попарно различные начальные действия, то необходимо не более s операций, чтобы сравнить начальные действия построенных графов и начальное действие проверяемой трассы ω.
3. Проверка равенства первого и последнего действия в ω с стоком и истоком графа G. Сложность: не превысит C, где C – некоторая константа.
4. Проверка $V' \subseteq V$. Сложность: не более un операций
5. Проверка достижимости вершин из V' из начальной вершины графа $G = (V, E)$. Сложность: не превысит $C'(n + e) + n^2$, где C' – некоторая константа.
6. Построение множества E', как построение отношения $\rightarrow_\omega$. Сложность: не более, чем $u + n^2$ операций.
7. Проверка $E' \subseteq E$. Сложность: не более en^2 операций.
8. Проверка связности графа G'. Сложность: не более $C''(n + e)$ операций, где C'' – некоторая константа.

Сложность шагов 1, 3 – 8, опираясь на доказательство Теоремы 3.2, составляет не более $C_1un + C_2en^2$. Таким образом, сложность поиска аномалий в некоторой трассе ω не превысит $s + C_1un + C_2en^2$, где C_1, C_2 – некоторые константы. ■

Если для задачи поиска аномалий построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 3.2 (Поиск аномалий в процессах, отличимых по начальному действию)**.** *Пусть s процессов имеют попарно различные начальные действия. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за $O(u)$.*

Рассматривается случай, когда не налагается условие на попарно различные начальные действия для каждого из s процессов, но есть возможность разделить лог L на части по одному уникальному для процесса действию. Стоит отметить, что таким образом обеспечивается [69] необходимое и достаточное условие существования системы различных представителей для теоремы Ф. Холла. Простейшим примером лога для этого случая для 2 процессов будет лог $L = \{ABA, AAC\}$, где действие B может встречаться только в процессе 1, действие C может встречаться только в процессе 2. Пусть $n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$. Пусть $m = \max_i |L_i| \to \infty$, L_i – часть лога L, соответствующая i-му процессу, $i = 1, \ldots, s$, k — максимальное количество повторов действия в логе L.

Теорема 3.5 (Отличимые процессы по некоторому действию)**.** *Пусть задан лог L для s процессов. Пусть каждый из s процессов имеет по одному известному действию A_i, $i = 1, \ldots, s$ такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей i-му процессу. Тогда сложность построения s конформных графов составляет $O\left(mn^2\right)$, $O\left(mn^3\right)$, $O\left(m\left(kn\right)^3\right)$ соответственно, в зависимости от свойств лога L.*

Доказательство. Если каждый из процессов имеет по одному известному заранее действию A_i, $i = 1, \ldots, s$, отличающему его от других, то за один проход по логу L можно получить разбиение по логам процессов L_i таких, что $L_i \cap L_j = \varnothing, i \neq j$, $i, j = 1, \ldots, s$.

Проход по логу не превышает $|L| \times |\omega_{max}|$, где ω_{max} – трасса лога L максимальной длины. Так как $|L| \leqslant sm$, а $|\omega_{max}| \leqslant kn$, то разбиение общего лога на логи процессов имеет оценку не превысит $smkn$, что не превышает оценок на построение s конформных графов зависимостей.

Так как часть лога L_i соответствует логу одного процесса, то в зависимости от свойств этой части лога будут справедливы условия одного из

Алгоритмов 3.1, 3.2, 3.3. Решение задачи построения s конформных графов может быть осуществлено за s запусков этих алгоритмов.

Так как $M \leqslant sm$, то справедливы следующие верхние оценки в зависимости от свойств лога L.

- Лог каждого из s процессов содержит только трассы, в которых каждое действие алфавита V_i встречается ровно по одному разу. Известно, что соотвествующие конформные логу L_i графы G_i не содержат циклов. Используя Лемму 3.1, s запусков Алгоритма 3.1 будет иметь оценку сложности $O\left(mn^2\right)$.
- Лог каждого из s процессов не содержит циклов. Используя Лемму 3.2, s запусков Алгоритма 3.2 будет иметь оценку сложности $O\left(mn^3\right)$.
- Пусть существует часть лога L_i, которая не удовлетворяет условиям Леммы 3.2. Используя Лемму 3.3, в худшем случае, s запусков Алгоритма 3.3 будет иметь оценку сложности $O\left(m\left(kn\right)^3\right)$, где k — максимальное количество повторов действия среди действий в логе.

■

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$,$i = 1,\ldots,s$, $u = |\omega| \to \infty$ — длина трассы ω.

Теорема 3.6 (Поиск аномалий в процессах, отличимых по некоторому действию)**.** *Пусть каждый из s процессов имеет по одному известному действию A_i, $i = 1,\ldots,s$ такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей i-му процессу. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за не более, чем $s + C_1un + C_2en^2$ операций, где C_1, C_2 — некоторые константы.*

Доказательство. Поиск аномалий в трассе ω возможно осуществить посредством проверки согласованности трассы с одним из построенных конформных графов зависимостей, то есть посредством применения Алгоритма 3.4.

Сложность шагов Алгоритма 3.4 при выполнении условий Теоремы 3.5 можно оценить следующим образом.

1. Инициализация V' множеством действий в $\boldsymbol{\omega}$, u операций.
2. Так как нам известны уникальные действия для процессов A_i, $i = 1, \ldots, s$, то достаточно одного прохода по трассе $\boldsymbol{\omega}$, чтобы понять, какому из действий A_i соответствует i-я буква трассы $\boldsymbol{\omega}$, тем самым выбрав нужный i-й конформный граф, сложность шага не более s.
3. Проверка равенства первого и последнего действия в $\boldsymbol{\omega}$ с стоком и истоком графа G. Сложность: не превысит C, где C – некоторая константа.
4. Проверка $V' \subseteq V$. Сложность: не более un операций
5. Проверка достижимости вершин из V' из начальной вершины графа $G = (V, E)$. Сложность: не превысит $C'(n + e) + n^2$, где C' – некоторая константа.
6. Построение множества E', как построение отношения $\rightarrow_{\omega}$. Сложность: не более, чем $u + n^2$ операций.
7. Проверка $E' \subseteq E$. Сложность: не более en^2 операций.
8. Проверка связности графа G'. Сложность: не более $C''(n + e)$ операций, где C'' – некоторая константа.

Сложность шагов 1, 3 – 8, опираясь на доказательство Теоремы 3.2, составляет не более $C_1un + C_2en^2$. Таким образом, поиск аномалий в некоторой трассе $\boldsymbol{\omega}$ может быть выполнен за не более, чем за $s + C_1un + C_2en^2$, где C_1, C_2 — некоторые константы. ■

Если для задачи поиска аномалий построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 3.3 (Поиск аномалий в процессах, отличимых по некоторому действию)**.** *Пусть каждый из s процессов имеет по одному известному действию A_i, $i = 1, \ldots, s$ такому, что оно не содержится в остальных процессах и содержится в каждой трассе, соответствующей i-му*

процессу. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в некоторой трассе ω может быть выполнен за $O(u)$.

Теорема 3.3 рассматривает случай уникальных для процессов подстрок длины 1, которые начинаются с первой буквы трасс лога L. Пусть $n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$. Пусть $m = \max_i |L_i| \to \infty, n = \max_i |V_i|, i = 1, \ldots, s$, k — максимальное количество повторов действия в логе L. Справедливо обобщение этой теоремы:

Теорема 3.7 (Отличимые процессы по конечной подстроке)**.** *Пусть задан лог L для s процессов. Пусть известно, что каждая из трасс лога, начиная с некоторого фиксированного номера i, содержит подстроки фиксированной длины r, каждая из которых может принадлежать только одному из s процессов. Тогда сложность построения s конформных графов составляет $O\left(mn^2\right)$, $O\left(mn^3\right)$, $O\left(m\left(kn\right)^3\right)$ соответственно, в зависимости от свойств лога L.*

Доказательство. Пусть $h : \Sigma^r \to \{0, \ldots s-1\}$ — идеальная хеш-функция [67]. Для разбиения лога L на части, соответствующие s процессам, необходимо для каждой трассы лога $A_1 \ldots A_q \in L$, $q \in \mathbb{N}$, вычислить $h(A_i, \ldots, A_{i+r})$. Согласно свойствам идеальной хеш-функции, такое вычисление в худшем случае имеет оценку сложности $O(1)$. Таким образом, сложность разбиения всего лога L на части составляет M вычислений функции h. Для хранения отображения подстрок длины r в номера процессов потребуется $O(sr)$ дополнительной памяти.

Так как часть лога L_i соответствует логу одного процесса, то в зависимости от свойств этой части лога будут справедливы условия одного из Алгоритмов 3.1, 3.2, 3.3. Решение задачи построения s конформных графов может быть осуществлено за s запусков этих алгоритмов.

Так как $M \leqslant sm$, то справедливы следующие верхние оценки в зависимости от свойств лога L.

– Лог каждого из s процессов содержит только трассы, в которых каждое действие алфавита V_i встречается ровно по одному разу. Известно, что соотвествующие конформные логу L_i графы G_i не

содержат циклов. Используя Лемму 3.1, s запусков Алгоритма 3.1 будет иметь оценку сложности $O\left(mn^2\right)$.

– Лог каждого из s процессов не содержит циклов. Используя Лемму 3.2, s запусков Алгоритма 3.2 будет иметь оценку сложности $O\left(mn^3\right)$.
– Пусть существует часть лога L_i, которая не удовлетворяет условиям Леммы 3.2. Используя Лемму 3.3, в худшем случае, s запусков Алгоритма 3.3 будет иметь оценку сложности $O\left(m\left(kn\right)^3\right)$, где k — максимальное количество повторов действия среди действий в логе.

■

Пусть $n = \max_i |V_i|$, $e = \max_i |E_i|$,$i = 1,\ldots,s$, $u = |\omega| \to \infty$ — длина трассы ω.

Теорема 3.8 (Поиск аномалий в процессах, отличимых по конечной подстроке)**.** *Пусть известно, что трасса ω, начиная с некоторого номера i содержит подстроку длины r, которая может принадлежать только одному из s процессов. Пусть для s процессов построено s конформных графов зависимостей, тогда поиск аномалий в трассе ω может быть выполнен за не более, чем $C_1un + C_2en^2$ операций, где C_1, C_2 — некоторые константы.*

Доказательство. Поиск аномалий в трассе ω возможно осуществить посредством проверки согласованности трассы с одним из построенных конформных графов зависимостей, то есть посредством применения Алгоритма 3.4.

Сложность шагов Алгоритма 3.4 при выполнении условий Теоремы 3.7 можно оценить следующим образом.

1. Инициализация V' множеством действий в ω, u операций.
2. Для определения нужного конформного графа необходимо использовать ту же идеальную хеш-функцию h, которая использовалась при построении моделей s процессов. Тогда сложность выполнения шага 1 составит $O(1)$.

3. Проверка равенства первого и последнего действия в ω с стоком и истоком графа G. Сложность: не превысит C, где C – некоторая константа.
4. Проверка $V' \subseteq V$. Сложность: не более un операций
5. Проверка достижимости вершин из V' из начальной вершины графа $G = (V, E)$. Сложность: не превысит $C'(n + e) + n^2$, где C' – некоторая константа.
6. Построение множества E', как построение отношения $\rightarrow_\omega$. Сложность: не более, чем $u + n^2$ операций.
7. Проверка $E' \subseteq E$. Сложность: не более en^2 операций.
8. Проверка связности графа G'. Сложность: не более $C''(n + e)$ операций, где C'' – некоторая константа.

Сложность шагов 1, 3 – 8, опираясь на доказательство Теоремы 3.2, составляет не более $C_1un + C_2en^2$. Таким образом, поиск аномалий в некоторой трассе ω может быть выполнен за не более, чем за $C_1un + C_2en^2$, где C_1, C_2 — некоторые константы. ■

Если для задачи поиска аномалий построены модели процессов в виде ациклических ориентированных графов, тогда параметры n, e будут константами и справедливо:

Следствие 3.4 (Поиск аномалий в процессах, отличимых по конечной подстроке)**.** *Пусть известно, что трасса* ω*, начиная с некоторого номера* i *содержит подстроку длины* r*, которая может принадлежать только одному из* s *процессов. Пусть для* s *процессов построено* s *конформных графов зависимостей, тогда поиск аномалий в трассе* ω *может быть выполнен за* $O(u)$*.*

3.4 Поиск аномалий с использованием модели процесса, построенной по логу, который содержит данные нескольких процессов внутри каждой из трасс

В этом разделе диссертации представлено решение задачи поиска аномалий с использованием модели процесса, которая была построена по логу L такому, что в логе L в одной трассе могут содержаться данные нескольких процессов.

Стоит заметить, что в Алгоритмах 3.1, 3.2, 3.3 после того, как были добавлены все зависимости из лога, будут удалены дуги, которые идут в обе стороны для пар действий. То есть, если согласно логу L существует зависимость $A \lesssim B$ и зависимость $B \lesssim A$, то дуги между вершинами, соответствующими действиям A и B, будут удалены. Если при этом действие A принадлежит процессу 1, а действие B принадлежит процессу 2, то это означает, что найдено условие, позволяющее отличать различные одновременно функционирующие процессы. Однако, это условие налагает ограничение на "полноту" лога, чтобы каждое из действий одного процесса было независимо от каждого из действий другого процесса.

Пусть $M = |L| \to \infty$, $n = \max_i |V_i| \to \infty$, $i = 1, \ldots, s$, $M >> n$.

Теорема 3.9 (Множество процессов в трассе без повторов действий)**.** *Пусть задан лог L для s процессов. В каждой из трасс лога L могут присутствовать действия от 1 до s процессов и в рамках одной трассы нет повторов действий. Пусть конформные графы для s процессов не содержат циклов. Пусть множества действий для процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$. Если для всех действий $A \in V_i, B \in V_j, i \neq j$, i, $j = 1, \ldots,$ s в логе L выполнено: $A \lesssim B$ и $B \lesssim A$, то сложность построения s конформных графов составляет $O\left(Mn^2\right)$.*

Доказательство. Так как множества действий для процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$, то каждое из действий в логе L может принадлежать только одному множеству V_i, $i = 1, \ldots, s$.

Если для некоторых действий $\forall A \in V_i, B \in V_j, i \neq j$, $i, j = 1, \ldots, s$ выполнено: $A \lesssim B$ и $B \lesssim A$, то пары действий из разных процессов будут независимы между собой. Так как такие пары действий независимы, то дуги между подграфами, соответствующими разным процессам, будут удалены на шаге удаления двунаправленных дуг (шаг 3 Алгоритма 3.1). Таким образом, при предположении, что лог одного процесса содержит достаточно информации о самом процессе, действия одного процесса будут принадлежать одной компоненте связности и образуется ровно s компонент связности.

В последующих шагах Алгоритма 3.1 не происходит добавления дуг. Поэтому далее Алгоритм 3.1 минимизирует построенные s компонент связности, каждая из которых задает модель одного из s процессов. Таким образом, Алгоритм 3.1 не требует модификации для данного расширения и работает корректно для лога, содержащего данные s процессов.

Так как в рамках каждой из трасс лога по условию нет повторов действий, и конформные графы для s процессов не имеют циклов, то будет справедлива оценка сложности Леммы 3.1 с поправкой на увеличившийся размер лога и составит $O\left(Mn^2\right)$. ■

Если построенные s конформных графов по Теореме 3.9 обладают свойствами из Теорем 3.1, 3.3, 3.5, 3.7, и некоторая трасса $\boldsymbol{\omega}$ содержит в себе исполнение только одного процесса, то для ответа на вопрос, является ли трасса $\boldsymbol{\omega}$ аномальной, могут быть применены Следствия 3.1, 3.2, 3.3, 3.4 соответственно.

Следующая теорема отвечает на вопрос о сложности поиска аномалий, когда и поступающая трасса $\boldsymbol{\omega}$ содержит в себе исполнения нескольких процессов, $u \to \infty$:

Теорема 3.10 (Поиск аномалий для трассы, содержащей исполнения множества процессов, имеющих различные множества действий)**.** *Пусть множества действий для s процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$. Пусть для s процессов построено s конформных графов зависимостей. Пусть в некоторой трассе $\boldsymbol{\omega}$ могут содержаться исполнения множества процессов. Тогда поиск аномалий*

в трассе ω *может быть выполнен за не более, чем* $s(s + C_1 un + C_2 en^2)$ *операций, где* C_1, C_2 *– некоторые константы.*

Доказательство. Для того, чтобы выполнялись условия Алгоритма 3.4, необходимо, чтобы трасса ω была бы разбита на подтрассы, каждая из которых содержала в себе исполнение только одного процесса.

Так как уже построено s конформных графов процессов, значит, известны множества V_i, $i = 1, \ldots, s$. Так как множества действий для s процессов не пересекаются, то есть $V_i \cap V_j = \varnothing$, при $i \neq j$, $i, j = 1, \ldots, s$, значит за один проход по трассе ω возможно однозначно определить какому из процессов принадлежит каждое из действий в этой трассе, сложность шага составляет ω. Пусть трасса ω содержит в себе действия из $s' \leqslant s$ процессов. То есть трасса ω может быть разбита на s' подтрасс ω_i, $i = 1, \ldots, s'$. Пусть максимальная длина подтрассы, соответствующей одному процессу, $p = \max |\omega_i|$ $i = 1, \ldots, s'$.

Теперь, когда построены трассы ω_i, $i = 1, \ldots, s'$, становится возможным применение Алгоритма 3.4. Трасса ω считается аномальной, если хотя бы одна из трасс ω_i является аномальной.

Сложность в худшем случае составляет u для разбиения трассы ω на подтрассы ω_i, $i = 1, \ldots, s'$, и s' вызовов Алгоритма 3.4. Так как выполнены условия Теоремы 3.1, то общая сложность поиска аномалий составляет не более $u + s'(s + C_0 pn + C_2 en^2)$ операций, где C_0, C_2 – некоторые константы. Так как $s' \leqslant s$, $p \leqslant u$, то сложность составит не более, чем $s(s + C_1 un + C_2 en^2)$, где C_1, C_2 – некоторые константы. ■

Если для задачи поиска аномалий построены модели процессов в виде ациклических ориентированных графов, тогда параметры s, n, e будут константами и справедливо:

Следствие 3.5 (Поиск аномалий для трассы, содержащей исполнения множества процессов, имеющих различные множества действий)**.** *Пусть множества действий для* s *процессов не пересекаются, то есть* $V_i \cap V_j = \varnothing$, *при* $i \neq j$, $i, j = 1, \ldots, s$. *Пусть для* s *процессов построено* s *конформных графов зависимостей. Пусть в некоторой трассе* ω *могут содержаться исполнения множества процессов. Тогда поиск аномалий в трассе* ω *может быть выполнен за* $O(u)$.

Можно рассчитать нижнюю границу количества действий в логе L по всем трассам для ограничений, налагаемых в Теореме 3.9. Пусть $R = \sum_{\omega \in L} |\omega|$ — общее количество действий в логе, $n_i = |V_i|, i = 1, \ldots, s$ — количество действий каждого из s процессов.

Следствие 3.6. *Пусть задан лог L, для него выполнены условия Леммы 3.1. Пусть в каждой из трасс лога L могут присутствовать действия от 1 до $s \geqslant 1$ процессов. Пусть для всех действий $A \in V_i, B \in V_j, i \neq j$, $i, j = 1, \ldots, s$ в логе L выполнено: $A \lesssim B$ и $B \lesssim A$. Тогда $|L| \geqslant 2$, $R \geqslant 2 \sum_{i=1}^{s} n_i$.*

Доказательство. Оценка следует из расчета количества действий в парах вида $A \lesssim B$ и $B \lesssim A$, где $A \in V_i, B \in V_j, j \neq i, i, j = 1, \ldots, s$. Так как в условиях Леммы 3.1 в одной трассе не может быть повторов действий, то трассы вида ABA и BAB невозможны. Поэтому для построения, например пары $A \lesssim B$ и $B \lesssim A$ потребуется 2 трассы AB и BA. Таким образом, чтобы отличить s процессов друг от друга потребуется, как минимум, 2 трассы в логе, где в первой трассе задается прямая зависимость между всеми действиями всех процессов, а во второй трассе обратная зависимость. ■

Примером достижения нижней оценки на количество трасс и действий из Следствия 3.6 являются те процессы, где зависимости каждого из процессов возможно уместить в одну трассу. Например, пусть $V_1 = \{A,B,C\}$ и процесс p_1 может содержать только трассу ABC, $V_2 = \{D,F\}$ и процесс p_2 может содержать только трассу DF. Тогда примером лога, в котором сохраняется зависимость внутри процессов, а между процессами действия независимы, является лог $\{ABCDF, DFABC\}$.

3.5 Выводы

В настоящей главе представлены решения задачи построения формальной модели множества процессов в виде ациклического ориентированного графа и подходы к решению задачи поиска аномалий с помощью построенных эталонных моделей процесса. Решение задача построения моделей процессов представлено для различных видов входных данных, лога исполнений процессов: когда трасса лога может содержать исполнение только одного процесса и когда трасса лога может содержать исполнения нескольких процессов. Решение задачи поиска аномалий в трассе продемонстрировано, как для случая, когда трасса может быть исполнением только одного процесса, так и для случая, когда проверяемая трасса может содержать исполнения нескольких процессов.

Найдены ограничения применимости алгоритмов для одного процесса и определены оценки сложности построения формальных моделей для множества процессов в зависимости от свойств лога. Для решения задачи поиска аномалий предложены алгоритмы с оценками сложности, линейными относительно длины трассы, для которой требуется дать ответ об аномальности, и квадратичными относительно максимально возможного количества действий и зависимостей в одном моделируемом процессе. Показано, что с помощью системы различных представителей возможно эффективно выделять траектории множества различных процессов. Также даны некоторые оценки сложности на дополнительно требуемую память и минимальный размер лога для случая, когда в рамках одной трассы встречаются данные множества процессов.

Заключение

Основные результаты работы заключаются в следующем.

1. Исследована возможность использования математических моделей в виде ациклических ориентированных графов (DAG) для решения задачи построения модели процесса и для решения задачи поиска аномалий.

   Показано, что для моделей, сформулированных в терминах ациклических ориентированных графов, удается успешно решать задачу поиска аномалий при условии наличия нескольких одновременно функционирующих процессов. В том числе продемонстрировано, что с помощью системы различных представителей можно эффективно выделять траектории множества различных процессов.
2. Получены временные оценки сложности построения моделей процесса при описании модели процесса с помощью ациклических ориентированных графов.

   При этом продемонстрировано, что в ряде случаев алгоритмы восстановления нескольких одновременно функционирующих процессов полностью повторяют их аналоги для одного функционирующего процесса, что позволяет использовать уже готовые оценки сложности для решения задачи поиска аномалий.
3. Получены временные оценки сложности выявления аномалий при описании модели процесса с помощью ациклических ориентированных графов, которые являются линейными относительно длины проверяемой трассы.
4. Показано, что использование формального аппарата в виде сетей Петри для решения задачи поиска аномалий затруднено тем фактом, что модель процесса не всегда удается однозначно восстановить.

   Налагая простые требования корректности, такие, как: свойство надежности; свойство, позволяющее обеспечить запрет на после-

довательное выполнение в сети конструкций синхронизации и выбора; полнота лога рабочего процесса для рассматриваемой сети; сохранение в сети отношения каузальности между переходами, если между соответствующими действиями в логе это отношение было выполнено — не гарантирован результат построения хоть какой-либо подходящей однозначной, корректной модели процесса. Продемонстрировано, что без перехода к более сложному классу сетей Петри класс технологических процессов, для которого возможно решить задачу поиска аномалий с использованием восстановленной моделью процесса, сильно ограничен.

5. Показано, что не каждая математическая модель описания реального процесса является подходящей для эффективного решения задач информационной безопасности. Тем самым продемонстрирован подход, позволяющий выбирать модели для конкретных задач информационной безопасности.

Список литературы

1. *Leno, V.* Robotic process mining: vision and challenges / V. Leno, A. Polyvyanyy, M. Dumas, M. La Rosa, F. Maggi // Business & Information Systems Engineering. — 2021. — Т. 63, № 3. — С. 301—314.
2. *Andrews, R.* Quality-informed semi-automated event log generation for process mining / R. Andrews, C. G. van Dun, M. T. Wynn, W. Kratsch, M. Röglinger, A. H. ter Hofstede // Decision Support Systems. — 2020. — Т. 132. — С. 113265.
3. *Pham, D.-L.* Process-Aware Enterprise Social Network Prediction and Experiment Using LSTM Neural Network Models / D.-L. Pham, H. Ahn, K.-S. Kim, K. P. Kim // IEEE Access. — 2021. — Т. 9. — С. 57922—57940.
4. *Shakya, S.* Process mining error detection for securing the IoT system / S. Shakya // Journal of ISMAC. — 2020. — Т. 2, № 03. — С. 147—153.
5. *Cerezo, R.* Process mining for self-regulated learning assessment in e-learning / R. Cerezo, A. Bogarin, M. Esteban, C. Romero // Journal of Computing in Higher Education. — 2020. — Т. 32, № 1. — С. 74—88.
6. *Grisold, T.* Using process mining to support theorizing about change in organizations / T. Grisold, B. Wurm, J. Mendling, J. Vom Brocke // Proceedings of the 53rd Hawaii International Conference on System Sciences. — 2020.
7. *Pika, A.* Privacy-preserving process mining in healthcare / A. Pika, M. T. Wynn, S. Budiono, A. H. Ter Hofstede, W. M. van der Aalst, H. A. Reijers // International journal of environmental research and public health. — 2020. — Т. 17, № 5. — С. 1612.
8. *Reinkemeyer, L.* Process Mining in Action: Principles, Use Cases and Outlook / L. Reinkemeyer. — Springer Nature, 2020.
9. *Sarno, R.* Anomaly detection in business processes using process mining and fuzzy association rule learning / R. Sarno, F. Sinaga, K. R. Sungkono // Journal of Big Data. — 2020. — Т. 7, № 1. — С. 1—19.

10. *Kusuma, G.* Process Mining of Disease Trajectories: A Feasibility Study / G. Kusuma, S. Sykes, C. McInerney, O. Johnson // Proceedings of the 13th International Joint Conference on Biomedical Engineering Systems and Technologies (Volume 5). Т. 5. — Science, Technology Publications. 2020. — С. 705—712.

11. *Agrawal, R.* Mining process models from workflow logs / R. Agrawal, D. Gunopulos, F. Leymann // International Conference on Extending Database Technology. — Springer. 1998. — С. 467—483.

12. *Cook, J. E.* Discovering models of software processes from event-based data / J. E. Cook, A. L. Wolf // ACM Transactions on Software Engineering and Methodology (TOSEM). — 1998. — Т. 7, № 3. — С. 215—249.

13. *Mannila, H.* Discovery of frequent episodes in event sequences / H. Mannila, H. Toivonen, A. I. Verkamo // Data mining and knowledge discovery. — 1997. — Т. 1, № 3. — С. 259—289.

14. *Schimm, G.* Process miner—a tool for mining process schemes from event-based data / G. Schimm // European Workshop on Logics in Artificial Intelligence. — Springer. 2002. — С. 525—528.

15. *Herbst, J.* Ein induktiver ansatz zur akquisition und adaption von workflow-modellen / J. Herbst. — Tenea Verlag Ltd., 2004.

16. *Van der Aalst, W.* Workflow mining: Discovering process models from event logs / W. Van der Aalst, T. Weijters, L. Maruster // IEEE transactions on knowledge and data engineering. — 2004. — Т. 16, № 9. — С. 1128—1142.

17. *Питерсон, Д.* Теория сетей Петри и моделирование систем / Д. Питерсон. — Мир, 1984.

18. *Gold, E. M.* Complexity of automaton identification from given data / E. M. Gold // Information and control. — 1978. — Т. 37, № 3. — С. 302—320.

19. *Miclet, L.* Grammatical inference / L. Miclet // Syntactic and Structural Pattern Recognition—Theory and Applications. — World Scientific, 1990. — С. 237—290.

20. *Van Der Aalst, W.* Workflow management: models, methods, and systems / W. Van Der Aalst, K. M. Van Hee, K. van Hee. — MIT press, 2004.

21. *Das, S.* A unified gradient-descent/clustering architecture for finite state machine induction / S. Das, M. C. Mozer // Advances in neural information processing systems. — Citeseer. 1994. — С. 19—26.

22. *Zeng, Z.* Learning finite state machines with self-clustering recurrent networks / Z. Zeng, R. M. Goodman, P. Smyth // Neural Computation. — 1993. — Т. 5, № 6. — С. 976—990.

23. *Agrawal, R.* Fast algorithms for mining association rules / R. Agrawal, R. Srikant [и др.] // Proc. 20th int. conf. very large data bases, VLDB. Т. 1215. — Citeseer. 1994. — С. 487—499.

24. *Agrawal, R.* Mining sequential patterns / R. Agrawal, R. Srikant // Proceedings of the eleventh international conference on data engineering. — IEEE. 1995. — С. 3—14.

25. *Agrawal, R.* Mining association rules between sets of items in large databases / R. Agrawal, T. Imieliński, A. Swami // Proceedings of the 1993 ACM SIGMOD international conference on Management of data. — 1993. — С. 207—216.

26. *Molnar, C.* Interpretable machine learning / C. Molnar. — Lulu. com, 2020.

27. *Arrieta, A. B.* Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI / A. B. Arrieta, N. Diaz-Rodriguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-López, D. Molina, R. Benjamins [и др.] // Information fusion. — 2020. — Т. 58. — С. 82—115.

28. *Rosemann, M.* A study of the evolution of the representational capabilities of process modeling grammars / M. Rosemann, J. Recker, M. Indulska, P. Green // International Conference on Advanced Information Systems Engineering. — Springer. 2006. — С. 447—461.

29. *Agarwal, R.* Object-oriented modeling with UML: a study of developers perceptions / R. Agarwal, A. P. Sinha // Communications of the ACM. — 2003. — T. 46, № 9. — C. 248—256.

30. *Sarshar, K.* Comparing the control-flow of epc and petri net from the end-user perspective / K. Sarshar, P. Loos // International conference on business process management. — Springer. 2005. — C. 434—439.

31. *Keller, G.* Semantische Prozeßmodellierung auf der Grundlage"Ereignisgesteuerter Prozeßketten (EPK)" / G. Keller, A.-W. Scheer, M. Nüttgens. — Inst. für Wirtschaftsinformatik, 1992.

32. *Recker, J.* Does it matter which process modelling language we teach or use? An experimental study on understanding process modelling languages without formal education / J. Recker, A. Dreiling // ACIS 2007 Proceedings. — 2007. — C. 45.

33. *Rozinat, A.* Conformance checking of processes based on monitoring real behavior / A. Rozinat, W. M. Van der Aalst // Information Systems. — 2008. — T. 33, № 1. — C. 64—95.

34. *Valiant, L. G.* A theory of the learnable / L. G. Valiant // Communications of the ACM. — 1984. — T. 27, № 11. — C. 1134—1142.

35. *Angluin, D.* Learning regular sets from queries and counterexamples / D. Angluin // Information and computation. — 1987. — T. 75, № 2. — C. 87—106.

36. *Ahonen, H.* Forming grammars for structured documents: an application of grammatical inference / H. Ahonen, H. Mannila, E. Nikunen // International Colloquium on Grammatical Inference. — Springer. 1994. — C. 153—167.

37. *Brazma, A.* Approaches to the automatic discovery of patterns in biosequences / A. Brazma, I. Jonassen, I. Eidhammer, D. Gilbert // Journal of computational biology. — 1998. — T. 5, № 2. — C. 279—305.

38. *Van der Aalst, W. M.* Discovering workflow performance models from timed logs / W. M. Van der Aalst, B. F. van Dongen // International Conference on Engineering and Employment of Cooperative Information Systems. — Springer. 2002. — C. 45—63.

39. *Van Dongen, B. F.* The ProM framework: A new era in process mining tool support / B. F. Van Dongen, A. K. A. de Medeiros, H. Verbeek, A. Weijters, W. M. van Der Aalst // International conference on application and theory of petri nets. — Springer. 2005. — С. 444—454.

40. *Mendling, J.* Seven process modeling guidelines (7PMG) / J. Mendling, H. A. Reijers, W. M. van der Aalst // Information and Software Technology. — 2010. — Т. 52, № 2. — С. 127—136.

41. *Rosemann, M.* Potential pitfalls of process modeling: part A / M. Rosemann // Business Process Management Journal. — 2006.

42. *Krogstie, J.* Process models representing knowledge for action: a revised quality framework / J. Krogstie, G. Sindre, H. Jørgensen // European Journal of Information Systems. — 2006. — Т. 15, № 1. — С. 91—102.

43. *Becker, J.* Guidelines of business process modeling / J. Becker, M. Rosemann, C. v. Uthmann // Business process management. — Springer, 2000. — С. 30—49.

44. *Iso, I.* 9126 Software product evaluation–quality characteristics and guidelines for their use / I. Iso, I. Std // ISO/IEC Standard. — 2001. — Т. 9126.

45. *Moody, D. L.* Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions / D. L. Moody // Data & Knowledge Engineering. — 2005. — Т. 55, № 3. — С. 243—276.

46. *Mendling, J.* Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness. Т. 6 / J. Mendling. — Springer Science & Business Media, 2008.

47. *Mendling, J.* Understanding the occurrence of errors in process models based on metrics / J. Mendling, G. Neumann, W. v. d. Aalst // OTM Confederated International Conferences"On the Move to Meaningful Internet Systems". — Springer. 2007. — С. 113—130.

48. *Kindler, E.* On the semantics of EPCs: Resolving the vicious circle / E. Kindler // Data & Knowledge Engineering. — 2006. — Т. 56, № 1. — С. 23—40.

49. *Quesada, A.* 3 methods to deal with outliers / A. Quesada. — URL: https://www.kdnuggets.com/2017/01/3-methods-deal-outliers.html.

50. *Maimon, O.* Data mining and knowledge discovery handbook / O. Maimon, L. Rokach. — Springer, 2005.

51. *Idreos, S.* Overview of data exploration techniques / S. Idreos, O. Papaemmanouil, S. Chaudhuri // Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data. — 2015. — С. 277—281.

52. *Ester, M.* A density-based algorithm for discovering clusters in large spatial databases with noise. / M. Ester, H.-P. Kriegel, J. Sander, X. Xu [и др.] // kdd. Т. 96. — 1996. — С. 226—231.

53. *Schubert, E.* DBSCAN revisited, revisited: why and how you should (still) use DBSCAN / E. Schubert, J. Sander, M. Ester, H. P. Kriegel, X. Xu // ACM Transactions on Database Systems (TODS). — 2017. — Т. 42, № 3. — С. 1—21.

54. sklearn.cluster.DBSCAN. — URL: https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html.

55. *Liu, F. T.* Isolation forest / F. T. Liu, K. M. Ting, Z.-H. Zhou // 2008 eighth ieee international conference on data mining. — IEEE. 2008. — С. 413—422.

56. sklearn.ensemble.IsolationForest. — URL: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.IsolationForest.html.

57. *Chandola, V.* Anomaly detection: A survey / V. Chandola, A. Banerjee, V. Kumar // ACM computing surveys (CSUR). — 2009. — Т. 41, № 3. — С. 1—58.

58. *Bezerra, F.* Algorithms for anomaly detection of traces in logs of process aware information systems / F. Bezerra, J. Wainer // Information Systems. — 2013. — Т. 38, № 1. — С. 33—44.

59. *Metz, C. E.* Basic principles of ROC analysis / C. E. Metz // Seminars in nuclear medicine. Т. 8. — Elsevier. 1978. — С. 283—298.

60. Лекция 5. Сети Петри. — URL: http://www.hpc-education.ru/files/lectures/2011/ershov/ershov_2011_lectures05.pdf.

61. *Reisig, W.* Lectures on petri nets i: basic models: advances in petri nets / W. Reisig, G. Rozenberg. — Springer Science & Business Media, 1998.

62. *Murata, T.* Petri nets: Properties, analysis and applications / T. Murata // Proceedings of the IEEE. — 1989. — Т. 77, № 4. — С. 541—580.

63. *Desel, J.* Free choice Petri nets / J. Desel, J. Esparza. — Cambridge university press, 1995.

64. *Van der Aalst, W. M.* The application of Petri nets to workflow management / W. M. Van der Aalst // Journal of circuits, systems, and computers. — 1998. — Т. 8, № 01. — С. 21—66.

65. Petri Nets. — URL: http://mlwiki.org/index.php/Petri_Nets.

66. *Aho, A. V.* The transitive reduction of a directed graph / A. V. Aho, M. R. Garey, J. D. Ullman // SIAM Journal on Computing. — 1972. — Т. 1, № 2. — С. 131—137.

67. *Cormen, T. H.* Introduction to algorithms / T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein. — MIT press, 2009.

Публикации автора по теме диссертации

В рецензируемых научных изданиях, рекомендованных для защиты в диссертационном совете МГУ по специальности 2.3.6

68. Построение моделей процесса с помощью простых сетей Петри / И. Ю. Терёхина, А. А. Грушо, Е. Е. Тимонина, С. Я. Шоргин // Системы и средства информатики. — 2020. — Т. 30, № 4. — С. 61—75.

69. Выявление аномалий с помощью метаданных / А. А. Грушо, Е. Е. Тимонина, Н. А. Грушо, И. Ю. Терёхина // Информатика и ее применения. — 2020. — Т. 14, № 3. — С. 76—80.

70. *Терёхина, И. Ю.* Выявление аномалий с использованием построенной модели процессов в виде ациклического ориентированного графа / И. Ю. Терёхина // Программная инженерия. — 2023. — Т. 14, № 6. — С. 285—291.

В сборниках трудов конференций

71. *Терёхина, И. Ю.* О поиске аномалий с использованием построенных моделей нескольких процессов / И. Ю. Терёхина // Ломоносовские чтения-2023: научная конференция, факультет ВМК МГУ имени М.В.Ломоносова. Тезисы докладов. — Москва : ООО МАКС Пресс, 2023. — С. 183—184.

72. *Terekhina, I. Y.* Выявление аномалий в условиях функционирующих процессов / I. Y. Terekhina // Proceedings of Academician O.B. Lupanov 14th International Scientific Seminar "Discrete Mathematics and Its Applications". — Keldysh Institute of Applied Mathematics, 2022. — С. 288—291.

В прочих изданиях

73. *Teryokhina, I.* Anomaly detection in several running processes / I. Teryokhina // International Journal of Open Information Technologies. — 2022. — Т. 10, № 1. — С. 21—27.