MODELING AND ANALYSIS OF INFORMATION SYSTEMS, VOL. 33, NO. 2, 2026
journal homepage: www.mais-journal.ru

THEORY OF COMPUTING

Graph-based Patterns in Inconsistent Declarative Process Models

A. N. Annenkov[1], R. A. Nesterov[1]

DOI: 10.18255/1818-1015-2026-2-176-205

[1]National Research University Higher School of Economics, Moscow, Russia

MSC2020: 68Q60, 68R10, 03B70
Research article
Full text in Russian

Received May 4, 2026
Revised May 22, 2026
Accepted May 27, 2026

Declarative process models are widely used in process mining to describe flexible process behavior through sets of constraints. However, models discovered automatically from event logs may contain inconsistent constraints, which can make them difficult to interpret and unusable for execution, conformance checking, or further analysis. Existing methods for consistency analysis either rely on automata-based constructions with high worst-case time complexity or use heuristics based on MIS (minimal inconsistent subsets) that do not provide a full formal characterization of the inconsistency patterns they detect. In this paper, we propose a graph-based approach to the inconsistency analysis for a restricted fragment of Declare process modeling language. We represent dependencies between constraints through the task entailment graph and characterize inconsistency by means of three structural witness types. Based on this characterization, we first detect candidate inconsistent subsets and then verify whether a candidate is a minimal inconsistent subset by dedicated verification procedures. In contrast to automata-based approaches, the proposed method avoids explicit automata products and relies instead on graph-based analysis and constructive trace arguments. We implement the proposed approach and evaluate it on real-life event logs, showing that it is practically feasible and achieves competitive runtime.

Keywords: process mining; declarative process models; inconsistency analysis; task entailment graph; minimal inconsistent subsets; structural witnesses

INFORMATION ABOUT THE AUTHORS

Annenkov, Aleksei N.	ORCID iD: 0009-0007-4682-5226. E-mail: alniannenkov@edu.hse.ru Research Assistant, Student
Nesterov, Roman A. (corresponding author)	ORCID iD: 0000-0002-4162-9070. E-mail: rnesterov@hse.ru PhD, Associate Professor, Head of the Laboratory

Funding: Basic Research Program at HSE University (HSE-BR-2025-024).

For citation: A. N. Annenkov and R. A. Nesterov, “Graph-based patterns in inconsistent declarative process models”, *Modeling and Analysis of Information Systems*, vol. 33, no. 2, pp. 176–205, 2026. DOI: 10.18255/1818-1015-2026-2-176-205.

МОДЕЛИРОВАНИЕ И АНАЛИЗ ИНФОРМАЦИОННЫХ СИСТЕМ, ТОМ 33, № 2, 2026
сайт журнала: www.mais-journal.ru

THEORY OF COMPUTING

Графовые паттерны в несогласованных декларативных моделях процессов

А. Н. Анненков[1], Р. А. Нестеров[1]

DOI: 10.18255/1818-1015-2026-2-176-205

[1]Национальный исследовательский университет «Высшая школа экономики», Москва, Россия

УДК 004.02
Научная статья
Полный текст на русском языке

Получена 4 мая 2026 г.
После доработки 22 мая 2026 г.
Принята к публикации 27 мая 2026 г.

Декларативные модели процессов широко используются в process mining для гибкого описания поведения процессов с помощью наборов ограничений. Однако модели, автоматически извлекаемые из журналов событий, могут содержать несогласованные ограничения, что затрудняет их интерпретацию и делает их непригодными для исполнения, проверки соответствия или дальнейшего анализа. Существующие методы анализа согласованности либо опираются на автоматные конструкции с высокой асимптотической сложностью в худшем случае, либо используют эвристики, ориентированные на минимальные несогласованные подмножества (MIS), которые не дают полного формального описания обнаруживаемых ими паттернов несогласованности. В данной работе предлагается графовый подход к анализу несогласованности для ограниченного фрагмента языка Declare. Мы представляем зависимости между ограничениями с помощью графа следования задач и характеризуем несогласованность посредством трёх типов структурных свидетелей. На основе этого описания мы сначала выявляем кандидаты в несогласованные подмножества, а затем проверяем, является ли кандидат минимальным несогласованным подмножеством, с помощью специализированных процедур верификации. В отличие от подходов, основанных на автоматах, предлагаемый метод избегает явного построения произведений автоматов и вместо этого опирается на графовый анализ и конструктивное построение трасс. Мы реализовали предложенный подход и провели его экспериментальную оценку на реальных журналах событий, показав, что он практически применим и обеспечивает конкурентоспособное время работы.

Ключевые слова: интеллектуальный анализ процессов; декларативные модели процессов; анализ несогласованности; граф следования задач; минимальные несогласованные подмножества; структурные свидетели

ИНФОРМАЦИЯ ОБ АВТОРАХ

Анненков, Алексей Николаевич	ORCID iD: 0009-0007-4682-5226. E-mail: alniannenkov@edu.hse.ru стажер-исследователь, студент
Нестеров, Роман Александрович (автор для корреспонденции)	ORCID iD: 0000-0002-4162-9070. E-mail: rnesterov@hse.ru канд. комп. наук, доцент, заведующий научно-учебной лабораторией

Финансирование: Программа фундаментальных исследований НИУ ВШЭ (HSE-BR-2025-024).

Для цитирования: A. N. Annenkov and R. A. Nesterov, “Graph-based patterns in inconsistent declarative process models”, *Modeling and Analysis of Information Systems*, vol. 33, no. 2, pp. 176–205, 2026. DOI: 10.18255/1818-1015-2026-2-176-205.

Введение

В последние годы область process mining активно развивается, объединяя методы анализа данных и теории процессов для изучения, мониторинга и улучшения реальных бизнес-процессов на основе журналов событий [1, 2]. Одной из центральных задач этой области является автоматическое построение модели процесса по записанным исполнениям. В зависимости от выбранной парадигмы моделирования методы обнаружения могут строить либо процедурные модели, явно описывающие допустимый поток управления, либо декларативные модели, задающие поведение с помощью ограничений [3, 4].

Декларативные модели представляют особый интерес в случаях, когда процесс обладает высокой вариативностью поведения и плохо описывается жёстко заданной последовательностью шагов. В таких моделях допустимое поведение задаётся не перечислением всех возможных сценариев, а множеством правил, которым должны удовлетворять трассы процесса. Поэтому декларативный подход особенно удобен для гибких и слабо структурированных процессов: часто естественнее не перечислять все допустимые пути исполнения явно, а неявно задавать множество допустимых трасс через набор ограничений [3, 4].

Язык Declare [3] является одним из наиболее известных языков декларативного моделирования процессов. При декларативном обнаружении модели Declare извлекаются из журналов событий путём выбора ограничений, удовлетворяющих порогам качества, таким как support, confidence или interest factor [5, 6]. Однако автоматическое извлечение декларативных моделей из журналов событий порождает важную проблему корректности получаемого результата. Автоматически построенные модели могут содержать несогласованные или избыточные ограничения [7, 8]. Несогласованные ограничения могут исключать все возможные трассы исполнения, то есть делать модель невыполнимой, тогда как избыточные ограничения увеличивают размер модели, не добавляя новой информации о поведении процесса. В обоих случаях полученная модель становится менее интерпретируемой и менее пригодной для исполнения, проверки соответствия, симуляции и дальнейшего анализа [7]. Хотя несогласованность и избыточность тесно связаны, в данной работе основное внимание уделяется несогласованности.

Для анализа согласованности декларативных моделей процессов был предложен ряд подходов. Методы, основанные на автоматах, дают строгие формальные гарантии и позволяют рассуждать о выполнимости, согласованности и избыточности [7]. Однако такие методы обычно опираются на вычислительно дорогие конструкции, в частности на произведения автоматов и проверки включения языков. В более общем случае задача выполнимости уже является вычислительно трудной для выразительных темпоральных логик и их вариантов на конечных трассах [9, 10]. С другой стороны, подходы, основанные на минимальных несогласованных подмножествах и мерах несогласованности, дают более непосредственное представление о конфликтующих фрагментах модели и ближе к задаче устранения несогласованности [8, 11]. Вместе с тем существующие MIS-ориентированные подходы для декларативных моделей процессов часто ограничены отдельными паттернами несогласованности и не всегда дают полное формальное описание обнаруживаемых структур [8]. Это оставляет место для методов, которые одновременно являются формально обоснованными и вычислительно применимыми для практически важных фрагментов Declare.

В данной работе развивается графовый подход к анализу несогласованности в ограниченном фрагменте Declare. Основная идея состоит в том, чтобы перейти от логического описания модели к её структурному представлению в виде графа следования задач и анализировать несогласованность через специальные графовые свидетели. На основе такого представления сначала выявляются кандидаты в несогласованные подмножества, а затем проверяется, является ли найденный кандидат минимальным несогласованным подмножеством. Получающаяся схема занимает промежуточное положение между общими автоматными методами, которые формально точны, но часто

вычислительно тяжелы, и эвристическими MIS-ориентированными подходами, которые структурно наглядны, но не всегда полностью формализованы.

Основные результаты работы состоят в следующем. Во-первых, вводится структурный графовый взгляд на несогласованность в декларативных моделях процессов и выделяются три типа свидетелей: позиционные, циклические и двухпутевые. Во-вторых, доказывается, что каждое минимальное несогласованное подмножество в рассматриваемом фрагменте содержит свидетель одного из этих типов. В-третьих, разрабатываются специализированные процедуры верификации, позволяющие проверить, является ли найденный кандидат минимальным несогласованным подмножеством, и доказывается их корректность. В-четвёртых, предложенный подход реализуется и экспериментально оценивается на реальных журналах событий с сопоставлением полученных результатов с существующими подходами.

Работа построена следующим образом. В разделе 1 вводятся необходимые предварительные сведения. В разделе 2 изучается несогласованность декларативных моделей процессов, вводятся типы структурных свидетелей и формулируется основной результат существования. В разделе 3 уточняется структура минимальных несогласованных подмножеств и приводятся соответствующие процедуры верификации. В разделе 4 представлены результаты экспериментальной оценки. В разделе 5 обсуждаются связанные работы. Наконец, последний раздел завершает статью и обозначает направления дальнейших исследований.

1. Предварительные сведения

В этом разделе вводятся основные понятия, используемые далее. Сначала рассматриваются декларативные ограничения языка Declare и их интерпретация на конечных трассах. Затем определяется декларативная модель процесса, язык, задаваемый моделью, и связанный с ней граф следования задач. В завершение формализуются понятия несогласованности и минимального несогласованного подмножества.

1.1. Декларативные ограничения

Пусть Σ — конечный алфавит событий, а Σ^* — множество всех конечных трасс над Σ. Декларативное ограничение задаёт логическое требование, ограничивающее множество трасс, допустимых для рассматриваемого процесса. В данной работе рассматриваются ограничения, выразимые в языке Declare, широко используемом языке декларативного моделирования процессов, основанном на параметризованных шаблонах ограничений [3, 12].

В литературе шаблоны Declare обычно представляются формулами темпоральной логики, в частности LTL или её конечно-трассового варианта; они также могут задаваться конечными автоматами или регулярными выражениями [3, 7, 12]. В данной работе мы в основном используем теоретико-языковой взгляд: каждый экземпляр шаблона рассматривается через множество конечных трасс, которые ему удовлетворяют.

Определение 1 (Декларативное ограничение)**.** Декларативным ограничением C над алфавитом Σ будем называть экземпляр шаблона Declare, интерпретируемый на конечных трассах. Его семантика задаётся языком

$$\mathcal{L}(C) \subseteq \Sigma^*.$$

Трасса $\sigma \in \Sigma^*$ удовлетворяет ограничению C, если $\sigma \in \mathcal{L}(C)$, в этом случае будем также писать $\sigma \models C$.

Для бинарных шаблонов Declare удобно различать событие-активацию и событие-цель.

Определение 2 (Событие-активация и событие-цель)**.** Пусть C — бинарное декларативное ограничение. Через $activate(C)$, $target(C)$ будем обозначать событие-активацию и событие-цель ограничения C соответственно.

Table 1. Representative Declare templates, and their activate and target events

Таблица 1. Шаблоны Declare, их события-активации и события-цели

Ограничение	Activate	Target	Объяснение
Existence(A)	⊤	A	A встречается хотя бы один раз
AtMostOnce(A)	⊤	A	A встречается не более одного раза
Init(A)	⊤	A	A — первое событие
Last(A)	⊤	A	A — последнее событие
Choice(A,B)	⊤	A или B	Встречается хотя бы одно из A, B
ExclusiveChoice(A,B)	⊤	A или B	Встречается ровно одно из A, B
CoExistence(A,B)	A или B	B или A	A и B либо оба есть, либо обоих нет
RespondedExistence(A,B)	A	B	Если есть A, то есть и B
Response(A,B)	A	B	После каждого A позже встречается B
AlternateResponse(A,B)	A	B	После A событие B должно наступить до нового A
ChainResponse(A,B)	A	B	Сразу после A идёт B
Precedence(A,B)	B	A	Каждому B предшествует A
AlternatePrecedence(A,B)	B	A	Каждому B предшествует A, между ними нет другого B
ChainPrecedence(A,B)	B	A	Непосредственно перед B стоит A
Succession(A,B)	A и B	B и A	Если A есть, должен быть B, и наоборот; порядок важен
AlternateSuccession(A,B)	A и B	B и A	A и B должны чередоваться, A перед B
ChainSuccession(A,B)	A и B	B и A	A и B должны идти строго подряд
NotCoExistence(A,B)	A или B	B или A	A и B не встречаются вместе
NotSuccession(A,B)	A	B	После A больше не должно быть B
NotChainSuccession(A,B)	A	B	Сразу после A не должно быть B

Для ограничения $Response(a, b)$ событием-активацией является a, а событием-целью — b. Для унарных шаблонов и некоторых бинарных шаблонов ограничение активно безусловно. В таких случаях будем писать $activate(C) = \top$, где $\top$ обозначает безусловную активацию с начала трассы.

В таблице 1 приведены рассматриваемые шаблоны ограничений, а также их события-активации и события-цели. Если у бинарного ограничения *activate* или *target* выражается через логическую связку «или», то далее будем учитывать две пары (*activate*, *target*): для ограничения $C(a, b)$ также рассматривается симметричный вариант $C(b, a)$.

Некоторые ограничения задают не только факт появления события-цели, но и его позицию относительно события-активации. Это мотивирует следующее определение.

Определение 3 (Позиционное ограничение)**.** Декларативное ограничение C называется *позиционным*, если его семантика фиксирует положение события-цели относительно события-активации в трассе.

В частности, позиционными являются шаблоны *Init*, *Last*, *ChainResponse*, *ChainPrecedence*, а также *ChainSuccession* и их отрицательные аналоги. Такие ограничения имеют семантику *directly follows*: положение события-цели задаётся относительно положения события-активации.

1.2. Декларативные модели процессов

Декларативная модель процесса задаётся конечным набором декларативных ограничений.

Определение 4 (Декларативная модель процесса)**.** Декларативной моделью процесса, или просто моделью, называется конечное множество декларативных ограничений

$$\mathcal{D} = \{C_1, \ldots, C_n\}.$$

Такой взгляд важен для дальнейшего анализа: поскольку модель является конечным множеством ограничений, к моделям естественным образом применимы стандартные операции над множествами, включая включение, объединение, пересечение, разность и выбор подмножеств. Более

общо, когда это необходимо, логические и теоретико-множественные отношения между ограничениями естественно переносятся на множества ограничений, а значит, и на декларативные модели.

Язык, задаваемый моделью, определяется стандартным образом.

Определение 5 (Язык, порождаемый декларативной моделью)**.** Пусть $\mathcal{D}$ — декларативная модель над алфавитом Σ. Языком, порождаемым моделью $\mathcal{D}$, называется множество всех трасс над Σ, удовлетворяющих всем ограничениям модели:

$$\mathcal{L}(\mathcal{D}) = \bigcap_{C \in \mathcal{D}} \mathcal{L}(C) = \{\sigma \in \Sigma^* \mid \sigma \models C \text{ для всех } C \in \mathcal{D}\}.$$

Таким образом, трасса принадлежит $\mathcal{L}(\mathcal{D})$ тогда и только тогда, когда она удовлетворяет каждому ограничению модели.

Для анализа зависимостей, порождаемых ограничениями, сопоставим модели ориентированный граф, называемый графом следования задач. Его вершинами служат события, а рёбра отражают зависимости между событиями-активациями и событиями-целями. Интуитивно ребро из a в b означает, что появление a влечёт требование, связанное с b.

Определение 6 (Граф следования задач)**.** Пусть $\mathcal{D}$ — декларативная модель над алфавитом Σ. *Графом следования задач* (*task entailment graph*) модели $\mathcal{D}$ называется ориентированный помеченный граф

$$G(\mathcal{D}) = (V, E, t),$$

где $V = \Sigma$, каждое бинарное ограничение $C \in \mathcal{D}$ порождает ребро

$$(activate(C), target(C)) \in E,$$

а $t : E \to T$ сопоставляет каждому ребру тип шаблона ограничения из множества типов T. Если выделенный символ $\top$ используется явно, он рассматривается как специальная исходная вершина, соответствующая безусловной активации.

Граф следования задач будет основным структурным объектом дальнейшего анализа. В качестве сквозного примера рассматривается декларативная модель процесса управления IT-релизом (см. рисунок 1). Она объединяет ограничения, возникающие в нескольких реалистичных фрагментах процесса: запуск сборки, проверка безопасности, согласование релиза и подготовка отката. Такой пример удобен тем, что соответствующий граф следования задач является связным и одновременно содержит свидетели всех трёх типов несогласованности, вводимых далее.

Здесь M обозначает *Merge Completed*, B — *Build Started*, S — *Security Scan Completed*, T — *Integration Tests Started*, A — *Approval Issued*, P — *Rollback Prepared*, а D — *Deployment Executed*.

Отметим также, что граф следования задач, представленный на рисунке 1, должен содержать дугу $D \to P$. Однако она соответствует тому же ограничению $NotCoExistence(P, D)$, что и уже имеющаяся дуга $P \to D$. Добавление дуги $D \to P$ не создаёт нового минимального несогласованного подмножества, а лишь даёт еще одно объяснение уже существующему двухпутевому свидетелю, как будет показано дальше. Этот рисунок нацелен на иллюстрацию трех типов минимальных несогласованных подмножеств, поэтому мы опустили эту дугу для повышения наглядности.

Ещё одно понятие, полезное для анализа, — ослабление ограничения.

Определение 7 (Ослабление ограничения)**.** Пусть C и $\mathcal{W}$ — два декларативных ограничения над Σ. Скажем, что $\mathcal{W}$ является *ослаблением* ограничения C, если каждое поведение, удовлетворяющее C, также удовлетворяет $\mathcal{W}$, то есть

$$\mathcal{L}(C) \subseteq \mathcal{L}(\mathcal{W}).$$

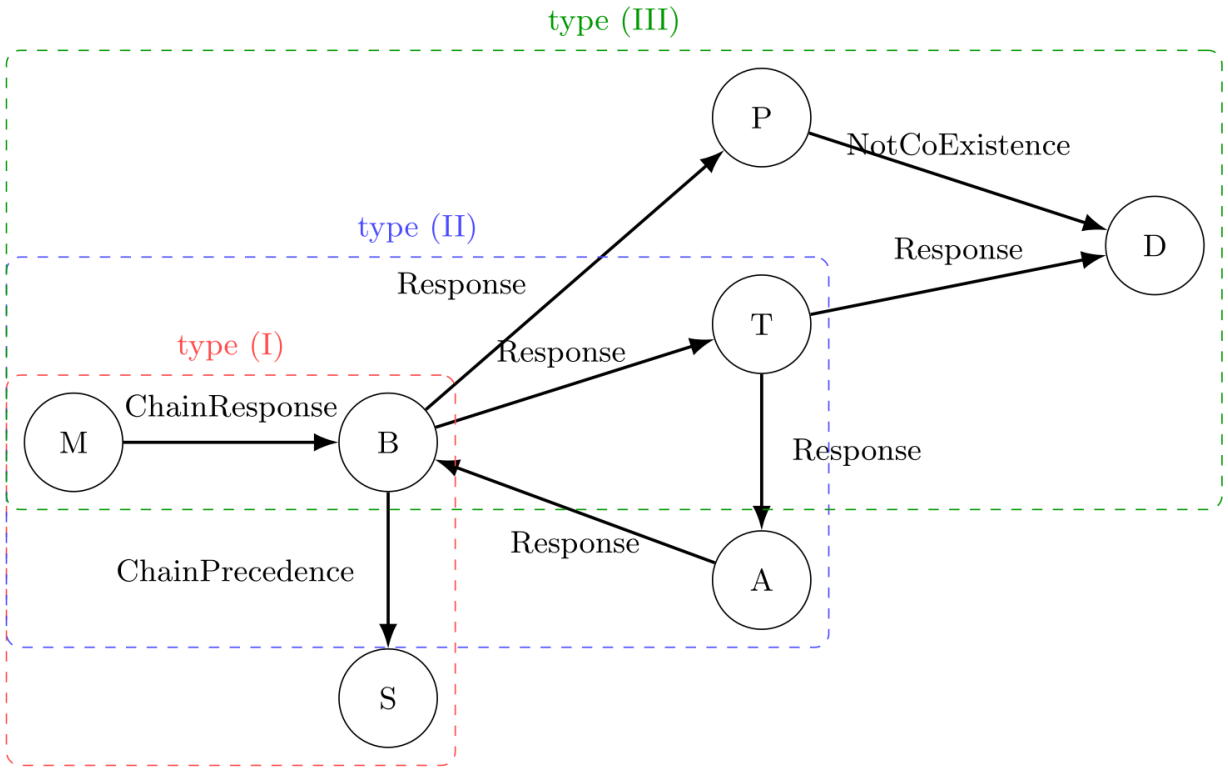

Fig. 1. Task entailment graph of the running declarative IT release-management model. Dashed regions indicate three overlapping minimal inconsistent subsets corresponding to the positional, cyclic, and two-path inconsistency patterns

Рис. 1. Граф следования задач сквозной декларативной модели процесса управления IT-релизом. Пунктирные области показывают три пересекающихся минимальных несогласованных подмножества, соответствующих позиционному, циклическому и двухпутевому типам несогласованности

Ограничение $Response(a, b)$ является ослаблением $ChainResponse(a, b)$, поскольку любая трасса, в которой b происходит сразу после каждого появления a, заведомо удовлетворяет более слабому требованию, что b происходит когда-нибудь позже после каждого появления a. В более общем виде рассматриваемые шаблоны задают естественную иерархию ослаблений. Эта иерархия показана на рисунке 2: более сильные и специальные ограничения расположены ниже на диаграмме, а более слабые — выше. Позиционные шаблоны выделены отдельной рамкой «positional templates», поскольку они играют особую роль в дальнейшем анализе несогласованности.

1.3. Несогласованность и минимальные несогласованные подмножества

Декларативная модель процесса называется несогласованной, если она не допускает ни одной удовлетворяющей трассы.

Определение 8 (Несогласованность)**.** Декларативная модель $\mathcal{D}$ называется *несогласованной*, если

$$\mathcal{L}(\mathcal{D}) = \varnothing.$$

Для структурного анализа несогласованности далее используется стандартное понятие минимального несогласованного подмножества.

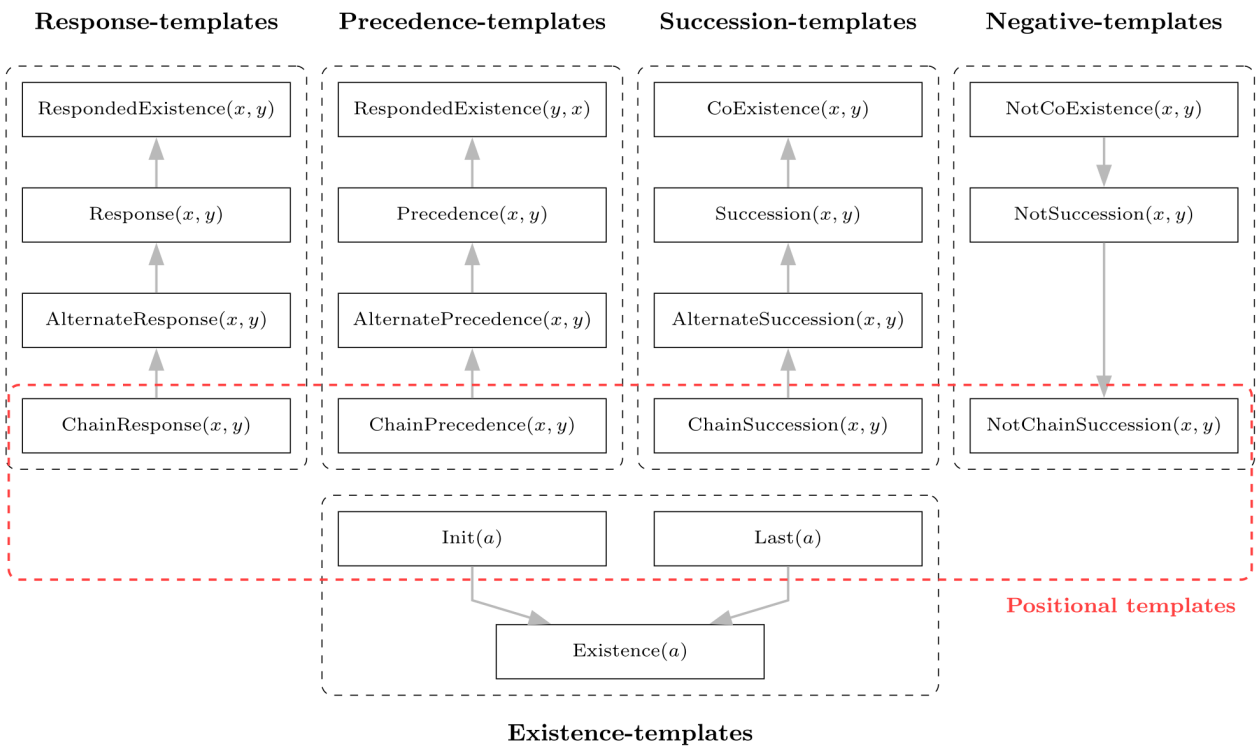

Fig. 2. Subsumption-based weakening relations between the considered Declare templates. Positional templates are highlighted by the special frame

Рис. 2. Отношения ослабления на основе включения языков между рассматриваемыми шаблонами Declare. Позиционные шаблоны выделены отдельной рамкой

Определение 9 (Минимальное несогласованное подмножество)**.** Пусть $\mathcal{D}$ — декларативная модель. Подмножество $\mathcal{M} \subseteq \mathcal{D}$ называется *минимальным несогласованным подмножеством* (Minimal Inconsistent Subset, MIS), если

$$\mathcal{L}(\mathcal{M}) = \varnothing \qquad \text{и} \qquad \forall \mathcal{M}' \subset \mathcal{M} : \ \mathcal{L}(\mathcal{M}') \neq \varnothing.$$

Таким образом, MIS — это минимальный по включению набор ограничений, который сам является несогласованным. Это понятие достаточно для локализации несогласованности внутри произвольной декларативной модели процесса: если модель несогласованна, то некоторое её подмножество является MIS, а ограничения, не входящие ни в одно такое подмножество, не относятся непосредственно к самому противоречию. Поэтому дальнейший анализ несогласованности сводится к анализу минимальных несогласованных подмножеств.

2. Несогласованность декларативной модели

В данном разделе мы изучаем структуру несогласованной модели через её граф следования. Использование графа позволяет формулировать ключевые паттерны несогласованности в терминах путей и циклов, что заметно упрощает чтение и последующие доказательства.

Раздел устроен следующим образом: сначала вводятся три базовых типа «свидетелей» противоречия в терминах графа, затем для каждого типа приводится иллюстрация достижимости, после чего доказывается теорема о существовании свидетеля в каждом MIS. Наконец, результаты переносятся с MIS на произвольную несогласованную модель.

2.1. Свидетели несогласованности

Перед введением трёх типов структурных свидетелей напомним, что среди шаблонов Declare, рассматриваемых в данной работе, особую роль играют позиционные ограничения. Они фиксируют непосредственное следование событий и поэтому могут порождать отдельный тип противоречия, исчезающий при замене позиционных ограничений на их непозиционные ослабления.

Определение 10 (Структурные типы свидетелей несогласованности)**.** Пусть $\mathcal{M}$ — некоторое MIS, и пусть $G(\mathcal{M}) = (\mathcal{A}, E, t)$ — граф следования, построенный по $\mathcal{M}$. Подмножество $\mathcal{M}_j \subseteq \mathcal{M}$ будем называть свидетелем несогласованности одного из следующих типов.

(I) **Позиционный свидетель.** В $\mathcal{M}_j$ присутствуют позиционные ограничения (см. определение 3), причём $\mathcal{M}_j$ является несогласованной, а модель *Weak* $\mathcal{M}_j$ — согласованной, где *Weak* $\mathcal{M}_j$ получается заменой каждого позиционного ограничения на его непозиционное ослабление.

(II) **Циклический свидетель.** В графе $G(\mathcal{M}_j)$ существует ориентированный цикл

$$x_1 \to x_2 \to \cdots \to x_k \to x_1,$$

причём каждое ребро цикла соответствует некоторому ограничению из $\mathcal{M}_j$. Другими словами, $\mathcal{M}_j$ содержит замкнутую цепочку обязательств, возвращающуюся к исходному событию.

(III) **Двухпутевой свидетель.** В графе $G(\mathcal{M}_j)$ имеются два различных ориентированных пути с общими началом и концом:

$$s \overset{p_1}{\rightsquigarrow} v \qquad \text{и} \qquad s \overset{p_2}{\rightsquigarrow} v,$$

где пути p_1 и p_2 различны хотя бы одним ребром. Иначе говоря, $\mathcal{M}_j$ порождает развилку обязательств, которая сходится в одном и том же событии-цели.

Далее покажем, что каждый из трёх типов достижим: это одновременно проясняет смысл паттернов и служит основой для дальнейших рассуждений. В качестве сквозной иллюстрации используется граф следования на рисунке 1.

Утверждение 1. Подмножества типа (I) существуют.

Доказательство. Достаточно рассмотреть подмножество

$$\mathcal{M} = \{\mathit{Existence}(M),\ \mathit{ChainResponse}(M, B),\ \mathit{ChainPrecedence}(S, B)\}.$$

На графе с рисунка 1 это соответствует левому нижнему фрагменту типа (I): событие M обязано появиться, после чего ограничение $\mathit{ChainResponse}(M, B)$ требует, чтобы B произошло непосредственно после M. Одновременно ограничение $\mathit{ChainPrecedence}(S, B)$ требует, чтобы B имело непосредственного предшественника S.

Если $M \neq S$, то одно и то же вхождение B не может одновременно непосредственно следовать за M и непосредственно иметь перед собой S. Следовательно, $\mathcal{M}$ является несогласованной.

При этом ослабленная версия

$$\mathit{Weak}\,\mathcal{M} = \{\mathit{Existence}(M),\ \mathit{AlternateResponse}(M, B),\ \mathit{AlternatePrecedence}(S, B)\}$$

является согласованной, поскольку допускает, например, трассу

$$\sigma = \langle M, S, B\rangle.$$

Следовательно, $\mathcal{M}$ реализует паттерн типа (I). Заметим также, что эта несогласованность обусловлена именно жёсткой позиционностью, то есть исчезает при переходе к ослабленным версиям directly-follows-ограничений. □

Утверждение 2. Подмножества типа (II) существуют.

Доказательство. Рассмотрим на графе с рисунка 1 подмножество

$$\mathcal{M} = \begin{Bmatrix} Existence(M),\ ChainResponse(M,B), \\ Response(B,T),\ Response(T,A), \\ Response(A,B) \end{Bmatrix}.$$

В графе $G(\mathcal{M})$ ограничения $Response(B,T)$, $Response(T,A)$ и $Response(A,B)$ образуют ориентированный цикл

$$B \to T \to A \ \to B.$$

При этом цикл действительно активируется, поскольку ограничение $Existence(M)$ совместно с $ChainResponse(M,B)$ гарантирует появление события B. После появления события B ограничение $Response(B,T)$ требует T, после появления T ограничение $Response(T,A)$ требует A, а после появления A ограничение $Response(A,B)$ снова требует новое вхождение B.

Тем самым возникает замкнутая система обязательств, которая на конечных трассах не может быть удовлетворена: любое удовлетворяющее выполнение должно было бы бесконечно порождать новые вхождения событий B, T и A. Следовательно, $\mathcal{M}$ является подмножеством типа (II). □

Утверждение 3. Подмножества типа (III) существуют.

Доказательство. Рассмотрим на графе с рисунка 1 подмножество

$$\mathcal{M} = \begin{Bmatrix} Existence(M),\ ChainResponse(M,B), \\ Response(B,T),\ Response(T,D), \\ Response(B,P),\ NotCoExistence(P,D) \end{Bmatrix}.$$

В графе $G(\mathcal{M})$ возникают две различные ветви, исходящие из общего старта B и ведущие к несовместимому завершению:

$$B \to T \to D \qquad \text{и} \qquad B \to P \dashv D.$$

Первая ветвь через T требует появления D, тогда как вторая ветвь требует появления P, после чего ограничение $NotCoExistence(P,D)$ запрещает совместное появление P и D. Иными словами, после запуска сборки модель одновременно навязывает две конкурирующие линии обязательств: одну к D, а другую к P, причём конечный результат этих двух ветвей несовместим.

Следовательно, данное подмножество реализует паттерн типа (III), и существует двухпутевой свидетель на одном и том же графе. □

Лемма 1. *Рассмотрим MIS $\mathcal{M}$ и ограничение $C(a,b) \in \mathcal{M}$ такое, что вершина b не имеет исходящих рёбер в $G(\mathcal{M} \setminus \{C(a,b)\})$, то есть b не является событием-активацией любого ограничения из $\mathcal{M} \setminus \{C(a,b)\}$. Тогда множество всех трасс, удовлетворяющих $\mathcal{M}' = \mathcal{M} \setminus \{C(a,b)\}$ и содержащих событие a, не пустое.*

Доказательство. Предположим противное: пусть таких трасс не существует. Тогда модель $\mathcal{M}'$ запрещает появление a, то есть из $\mathcal{M}'$ следует $Absence(a)$. Но тогда добавление ограничения $C(a,b)$ не может изменить язык модели, поскольку $C(a,b)$ активируется только при появлении a, а любые трассы с a уже исключены $\mathcal{M}'$. Следовательно,

$$\mathcal{L}(\mathcal{M}) = \mathcal{L}(\mathcal{M}'),$$

то есть $\mathcal{M}$ содержит собственное несогласованное подмножество, что противоречит минимальности $\mathcal{M}$ как MIS. □

Теорема 1 (О наличии свидетеля). *Для любой MIS $\mathcal{M}$ существует её подмножество, для которого выполнен хотя бы один из типов (I), (II), (III). Такое подмножество будем называть* свидетелем *противоречия.*

Доказательство. Типы (I) и (II) достижимы по утверждениям 1 и 2. Остаётся доказать импликацию

$$\overline{(I)} \wedge \overline{(II)} \implies (III).$$

Пусть $\mathcal{M}$ — произвольное MIS, а также выполнены $\overline{(I)}$ и $\overline{(II)}$. Из $\overline{(I)}$ следует, что все позиционные ограничения (см. определение 3) можно заменить их ослаблениями, не устраняя противоречие, поэтому далее считаем, что в $\mathcal{M}$ позиционных ограничений нет. Рассмотрим граф следования $G(\mathcal{M}) = (\mathcal{A}, E, t)$, построенный по $\mathcal{M}$.

Из $\overline{(II)}$ следует, что в $G(\mathcal{M})$ нет ориентированных циклов. Следовательно, в $G(\mathcal{M})$ существует вершина $b \in \mathcal{A}$, для которой нет исходящих рёбер, то есть b не является событием-активацией ни одного ограничения, представленного ребром в графе. Так как $\mathcal{M}$ — MIS и, в частности, $\mathcal{M} \neq \emptyset$, найдётся хотя бы одно ребро, входящее в b и выходящее из некоторого a; зафиксируем соответствующее ограничение и обозначим его как $C(a, b) \in \mathcal{M}$, где $activate(C) = a$ и $target(C) = b$. Положим

$$\mathcal{M}' = \mathcal{M} \setminus \{C(a, b)\}.$$

По лемме 1 множество трасс, удовлетворяющих $\mathcal{M}'$ и содержащих событие a, не является пустым. Возьмём любую такую трассу $\sigma \models \mathcal{M}'$ с $a \in \sigma$. Для ограничения $C(a, b)$ рассмотрим трассу σ_{fix}, полученную из σ только вставками или удалениями событий b так, чтобы $\sigma_{fix} \models C(a, b)$.

Так как $\mathcal{M}$ — MIS, то $\sigma_{fix} \not\models \mathcal{M}$. При этом $\sigma \models \mathcal{M}'$, а переход $\sigma \rightsquigarrow \sigma_{fix}$ меняет только вхождения b. Значит, существует ограничение $D \in \mathcal{M}'$, нарушенное на σ_{fix}, и это нарушение вызвано именно изменением вхождений b. Поскольку по выбору b в графе нет рёбер, исходящих из b, событие b не является событием-активацией ни одного из ограничений в $\mathcal{M}'$. Следовательно, добавление или удаление b не может породить новых ограничений, активируемых b, а потому нарушенное ограничение D обязательно имеет b в роли целевого события. Итак, существует ограничение $C'(c, b) \in \mathcal{M}'$ такое, что $target(C') = b$ и $activate(C') = c$.

Таким образом, в графе $G(\mathcal{M})$ имеются по крайней мере два различных входящих в b ребра: (a, b) и (c, b).

Далее рассмотрим ребро (a, b) и будем идти назад по ограничениям, восстанавливая путь: если текущая вершина x является целевой вершиной некоторого ограничения $K(u, x) \in \mathcal{M}$, то добавим ребро (u, x) и перейдём к u. Так как в $G(\mathcal{M})$ нет циклов, этот процесс обязательно завершится в вершине s, которая не является целевой ни одного ограничения из $\mathcal{M}$. В результате получаем ориентированный путь

$$P_1 : \quad s \rightsquigarrow a \to b,$$

все рёбра которого соответствуют ограничениям из $\mathcal{M}$. Аналогично, начиная с ребра (c, b), получаем путь

$$P_2 : \quad r \rightsquigarrow c \to b,$$

где r также не является целевой вершиной ни одного ограничения из $\mathcal{M}$.

Если $s = r$, то пути P_1 и P_2 имеют общий конец b и общий старт s, но различаются хотя бы последним ребром. Тогда, положив $\mathcal{M}_j^1$ равным множеству ограничений, соответствующих рёбрам пути P_1, а $\mathcal{M}_j^2$ — множеству ограничений, соответствующих рёбрам пути P_2, получаем ситуацию типа (III): две различные цепочки обязательств с совпадающими началом и концом.

Осталось исключить случай $s \neq r$. Без ограничения общности рассмотрим путь P_1 и его первое ребро (s, x), соответствующее некоторому ограничению $K(s, x) \in \mathcal{M}$. Положим

$$\widetilde{\mathcal{M}} = \mathcal{M} \setminus \{K(s, x)\}.$$

Так как $\mathcal{M} - \text{MIS}$, подмодель $\widetilde{\mathcal{M}}$ является согласованной, то есть, существует трасса $\tau \models \widetilde{\mathcal{M}}$.

Построим трассу τ' из τ удалением всех вхождений события s. Покажем, что $\tau' \models \mathcal{M}$, что даст противоречие с несогласованностью $\mathcal{M}$.

Действительно:

1. Так как s не является целевым событием ни одного ограничения из $\mathcal{M}$, удаление s не может нарушить ограничение, которое требовало бы появления s.
2. Отсутствие позиционных ограничений гарантирует, что удаление s не затрагивает требований непосредственного следования.
3. Любое ограничение из $\widetilde{\mathcal{M}}$, в котором s выступало событием-активацией, после удаления s становится только легче выполнить; ограничения, не содержащие s, сохраняют истинность при удалении несвязанных событий. Следовательно, $\tau' \models \widetilde{\mathcal{M}}$.
4. Наконец, удалённое ограничение $K(s, x)$ также выполняется на τ' тривиально, поскольку событие-активация s в τ' отсутствует, а значит, обязательства, порождаемые $K(s, x)$, не активируются.

Итак, $\tau' \models \widetilde{\mathcal{M}}$ и $\tau' \models K(s, x)$, откуда $\tau' \models \mathcal{M} = \widetilde{\mathcal{M}} \cup \{K(s, x)\}$, что противоречит тому, что $\mathcal{M}$ является несогласованной.

Полученное противоречие показывает, что случай $s \neq r$ невозможен. Значит, обязательно $s = r$, и, как было показано выше, выполняется тип (III). □

Утверждение 4. Внутри любой несогласованной модели $\mathcal{D}$ можно выделить MIS.

Доказательство. Рассмотрим множество всех подмоделей $\mathcal{D}_i \subseteq \mathcal{D}$, для которых $\mathcal{L}(\mathcal{D}_i) = \emptyset$, и выберем среди них подмодель, минимальную относительно вложения. По определению, это будет MIS: она является несогласованной, а любое её собственное подмножество согласовано. □

Теорема 2 (О структуре несогласованной модели)**.** *Рассмотрим произвольную декларативную модель $\mathcal{D}$. Если она является несогласованной, то в ней существует подмножество ограничений $\mathcal{M}_j \subseteq \mathcal{D}$, являющееся свидетелем одного из типов (I), (II) или (III).*

Доказательство. По утверждению 4 внутри $\mathcal{D}$ существует MIS $\mathcal{M} \subseteq \mathcal{D}$. По теореме 1 внутри $\mathcal{M}$ существует свидетель одного из типов (I)–(III). Так как $\mathcal{M} \subseteq \mathcal{D}$, этот свидетель также является подмножеством исходной модели $\mathcal{D}$. □

Итак, внутри каждой несогласованной модели содержится MIS, а в каждом MIS существует свидетель противоречия одного из типов (I)–(III). Следовательно, несогласованность произвольной модели обусловлена наличием в ней по крайней мере одного из описанных графовых паттернов.

2.2. Обнаружение потенциальных MIS и соответствующие графовые структуры

Теорема 2 утверждает, что если декларативная модель $\mathcal{D}$ является несогласованной, то она содержит подмножество ограничений одного из трёх структурных типов (I)–(III). Следовательно, для обнаружения потенциальных несогласованностей достаточно выделять подмножества, удовлетворяющие соответствующим графовым критериям. Это приводит к трёхфазной процедуре обнаружения кандидатов в MIS.

2.2.1. Фаза 1: позиционные кандидаты

Противоречия типа (I) возникают из-за несовместимости позиционных ограничений, навязывающих требования непосредственного следования событий. Такие противоречия исчезают после ослабления или удаления одного из конфликтующих позиционных ограничений, поэтому позиционная часть модели рассматривается первой и изолированно.

Рассмотрим множество событий Σ и введём специальную вершину $\top$, интерпретируемую как безусловную активацию. По позиционным ограничениям строится ориентированный граф прямого следования

$$G_{df} = (V_{df}, E_{df}), \qquad V_{df} = \Sigma \cup \{\top\},$$

где дуга $(u, v) \in E_{df}$ означает, что модель требует, чтобы событие v непосредственно следовало за событием u.

Дуги добавляются так: $Init(a)$ добавляет дугу $(\top, a)$, $Last(a)$ — дугу $(a, \top)$, $ChainResponse(a, b)$ и $ChainPrecedence(a, b)$ — дугу (a, b), а $ChainSuccession(a, b)$ добавляет дуги (a, b) и (b, a). Запрещающие позиционные ограничения $NotChainSuccession(a, b)$ сохраняются как множество запрещённых дуг:

$$F_{df} = \{(a, b) \mid NotChainSuccession(a, b) \in \mathcal{D}\}.$$

Система обязательных directly-follows требований должна быть функциональной: у каждой вершины может быть не более одного обязательного непосредственного предшественника и не более одного обязательного непосредственного последователя. Нарушение этого условия, наличие дуги, которая одновременно является обязательной и запрещённой, а также цикл, принудительно проходящий через $\top$, сразу дают кандидат типа (I).

Если локального конфликта не обнаружено, то G_{df} является дизъюнктным объединением ориентированных цепочек и циклов. В этом случае алфавит можно укрупнить, объединяя события, чья взаимная позиция жёстко фиксирована позиционными ограничениями. Этот шаг нормализации заменяет группы непосредственно соседних событий блоками и удаляет позиционный фрагмент перед последующими фазами анализа.

2.2.2. Фаза 2: циклические кандидаты

После нормализации позиционной части рассматривается граф следования задач $G = (V, E)$, вершины которого соответствуют событиям или блокам, а дуги — непозиционным ограничениям модели.

Противоречие типа (II) определяется не только самим ориентированным циклом: оно может зависеть и от пути, ведущего в этот цикл. Такой путь играет роль предцикла, активирующего циклическую часть несогласованности. Поэтому кандидаты типа (II) включают не только ограничения, порождающие цикл, но и ограничения, порождающие максимальные пути, ведущие к нему.

Для выделения таких структур сначала вычисляются компоненты сильной связности (SCC) графа G. Каждая SCC размера не меньше двух содержит ориентированный цикл, а SCC из одной вершины также задаёт цикл, если содержит петлю. Затем граф конденсируется в ациклический граф G^*, вершины которого соответствуют SCC. В этом ациклическом графе предциклы являются путями, заканчивающимися в циклических SCC.

Вместо перечисления всех возможных таких путей, число которых может быть экспоненциальным, достаточно сохранять только максимальных предшественников для каждой циклической SCC. Объединение ограничений, порождающих циклическую SCC, и соответствующего максимального предцикла образует кандидат типа (II).

2.2.3. Фаза 3: двухпутевые кандидаты

Противоречия типа (III) соответствуют существованию двух различных ориентированных путей с общим началом и общим концом. В терминах графа G для фиксированных вершин s и t это означает существование двух различных путей из s в t.

Прямое перечисление всех таких пар путей может быть экспоненциальным. Поэтому вместо перечисления всех путей используется структурный критерий: для заданной пары (s, t) два рёберно-непересекающихся пути из s в t существуют тогда и только тогда, когда минимальный s–t-разрез

имеет мощность не меньше двух. По теореме Менгера это эквивалентно тому, что значение максимального потока из s в t в графе с единичными пропускными способностями на рёбрах не меньше двух.

Таким образом, для каждой релевантной пары (s, t) проверяется, не меньше ли двух значение максимального потока. При положительном ответе восстанавливаются два соответствующих пути, а объединение ограничений, порождающих их рёбра, образует кандидат типа (III).

В результате трёхфазная процедура возвращает семейство подмножеств-кандидатов, каждое из которых может содержать свидетель несогласованности. По теореме 2 каждая несогласованность должна возникать внутри хотя бы одного такого кандидата. Поэтому поиск несогласованности можно ограничить этими структурными кандидатами, а не рассматривать все подмножества модели.

В следующем разделе эти кандидаты проверяются более точно, чтобы определить, являются ли они минимальными несогласованными подмножествами.

3. Структурные свойства и верификация кандидатов на MIS

В предыдущем разделе был описан алгоритм, выделяющий в несогласованной модели подмножества, являющиеся кандидатами типов (I)–(III). Однако сам факт того, что подмножество имеет один из этих структурных типов, ещё не означает, что оно является минимальным несогласованным множеством.

Поэтому далее мы уточняем внутреннюю структуру MIS каждого из типов (I)–(III), а затем используем полученные свойства для построения точного алгоритма проверки кандидата на MIS.

3.1. Структурные свойства MIS

Перед тем как перейти к алгоритму проверки кандидата на MIS, зафиксируем принципиальное деление ограничений на два класса, которое лежит в основе всего дальнейшего анализа.

Определение 11 (Обязывающие и запрещающие ограничения)**.** Декларативное ограничение $C(a, b)$ ($activate(C) = a$, $target(C) = b$) называется *обязывающим* (obligatory), если его семантика требует появления события b при наступлении события a.

Ограничение называется *запрещающим* (restrictive), если его семантика запрещает появление события b либо запрещает его появление в некотором контексте, порядке или позиции при наступлении события a.

Ограничения семейств *Response*, *Precedence*, *Succession*, *RespondedExistence* и *CoExistence* далее рассматриваются как обязывающие, а ограничения *NotCoExistence*, *NotSuccession* и *NotChainSuccession* — как запрещающие.

Утверждение 5 (Дихотомия нарушений)**.** Пусть $M = M_{obl} \sqcup M_{res}$ — декларативная модель, разбитая на обязывающие и запрещающие ограничения. Для трассы $\sigma \in A^*$ обозначим:

$$Miss(\sigma) \;\Leftrightarrow\; \exists\, C \in M_{obl} : \sigma \not\models C, \qquad Bad(\sigma) \;\Leftrightarrow\; \exists\, C \in M_{res} : \sigma \not\models C.$$

Тогда

$$L(M) = \varnothing \;\Leftrightarrow\; \forall\, \sigma \in A^* :\; Miss(\sigma) \vee Bad(\sigma).$$

Доказательство. По определению языка модели

$$\sigma \in L(M) \;\Leftrightarrow\; \forall C \in M :\; \sigma \models C.$$

Следовательно,

$$\sigma \notin L(M) \;\Leftrightarrow\; \exists\, C \in M :\; \sigma \not\models C.$$

Поскольку каждое ограничение принадлежит ровно одному из классов M_{obl} или M_{res}, существование нарушенного ограничения равносильно условию $Miss(\sigma) \vee Bad(\sigma)$. Условие $L(M) = \varnothing$ означает, что ни одна трасса не принадлежит языку модели, то есть $\forall \sigma \in A^* : \ \sigma \notin L(M)$, что и даёт требуемое утверждение. □

Утверждение 5 допускает следующую конструктивную интерпретацию. Рассмотрим стратегию последовательного построения трассы, при которой активированные обязательства устраняются немедленно: как только выполненное событие b активирует требование появления c, трасса расширяется именно событием c. При такой стратегии нарушение типа *Miss* — невыполненное обязательство в конце трассы — невозможно по построению. Следовательно, несогласованность модели может проявиться только через локальный конфликт: очередное ожидаемое событие оказывается одновременно запрещённым. Именно этот принцип будет использоваться далее при построении канонических трасс.

В этом разделе мы уточняем событийную структуру MIS каждого из типов (I)–(III), выделенных в теореме 1. Полученные структурные леммы служат основанием для алгоритмов следующего подраздела: они гарантируют, что канонические трассы, строящиеся этими алгоритмами, действительно охватывают все возможные способы удовлетворить множество ограничений.

3.1.1. Структура MIS типа (I)

MIS типа (I) порождены позиционными ограничениями и обнаруживаются уже в фазе 1 алгоритма поиска кандидатов через анализ графа G_{df}. Их проверка на минимальность тривиальна: малое число вовлечённых ограничений позволяет перебрать все собственные подмножества за линейное время.

3.1.2. Структура MIS типа (III)

Рассмотрим кандидата типа (III): потенциальное MIS $\mathcal{M}$ с двумя путями

$$p_1 : a \ \to b \to \cdots \to y \to x, \qquad p_2 : a \ \to c \to \cdots \to z \to x$$

в графе следования $G(\mathcal{M})$.

Лемма 2 (Позиция запрещающих ограничений в MIS типа (III))**.** *Пусть $\mathcal{M}$ — MIS типа (III) с путями p_1 и p_2. Тогда запрещающее ограничение может находиться только на последнем месте каждого из путей, то есть быть равным $C(y, x)$ или $C(z, x)$.*

Доказательство. Предположим, что в пути p_1 существует запрещающее ограничение $C_1(n, m)$, причём $m \neq x$, то есть это ограничение не является последним на пути. По построению пути все ограничения от a до n включительно являются обязывающими.

Рассмотрим трассу

$$\sigma = \left\langle \begin{array}{l} \text{все события из } p_1 \text{ от } a \ \text{ до } n \text{ включительно,} \\ \text{все события из } p_2, \text{ кроме } a, \ \text{от } c \text{ до } k \text{ включительно} \end{array} \right\rangle,$$

где $C_2(k, l)$ — первое запрещающее ограничение пути p_2 (если такого нет, полагаем $k = z$).

По построению все обязывающие ограничения обоих путей до соответствующих точек выполнены, поскольку события добавлены в порядке, согласованном с требованиями этих ограничений. Ограничение $C_1(n, m)$ выполнено, так как событие m в трассе σ отсутствует. Аналогично, при наличии ограничения $C_2(k, l)$ оно также выполняется, поскольку его событие-цель в трассу не добавляется. Следовательно, $\sigma \models \mathcal{M}$, что противоречит несогласованности $\mathcal{M}$.

Аналогичное рассуждение применяется к пути p_2. Следовательно, запрещающее ограничение в MIS типа (III) может стоять только в конце пути. □

Лемма 3 (Дихотомия концевых ограничений в MIS типа (III)). *Пусть $\mathcal{M}$ — MIS типа (III). Тогда ровно одно из двух концевых ограничений $C(y, x)$ и $C(z, x)$ является запрещающим.*

Доказательство. Сначала предположим, что оба концевых ограничения запрещающие. Тогда рассмотрим трассу, содержащую все события путей p_1 и p_2, кроме события x. Все внутренние обязывающие ограничения обоих путей выполнены, так как их события-цели входят в трассу. Оба концевых запрещающих ограничения выполнены тривиально, поскольку x в трассе отсутствует. Следовательно, $\mathcal{M}$ имеет удовлетворяющую трассу, что противоречит его несогласованности.

Теперь предположим, что оба концевых ограничения обязывающие. Тогда рассмотрим трассу, в которой сначала идёт событие a, затем все промежуточные события пути p_1, затем все промежуточные события пути p_2, и в конце добавляется x. По построению все внутренние обязывающие ограничения обоих путей выполнены. Кроме того, концевые ограничения $C(y, x)$ и $C(z, x)$ также выполнены, поскольку x расположено после y и z. Следовательно, и в этом случае $\mathcal{M}$ имеет удовлетворяющую трассу, что снова противоречит несогласованности.

Таким образом, оба рассмотренных симметричных случая невозможны. Следовательно, ровно одно из ограничений $C(y, x)$ и $C(z, x)$ должно быть запрещающим. □

3.1.3. Структура MIS типа (II)

Рассмотрим теперь кандидатов типа (II): они содержат ориентированный цикл в графе следования, возможно, с предциклом, то есть путём, ведущим к точке входа в цикл.

Лемма 4 (Структура MIS типа (II) без предцикла). *Пусть $\mathcal{M}$ — MIS типа (II), и граф следования $G(\mathcal{M})$ содержит цикл*

$$a \to b \to \cdots \to a$$

без предцикла. Тогда:

1. *Все ограничения $\mathcal{M}$ являются обязывающими.*
2. *В $\mathcal{M}$ нет ограничений RespondedExistence.*
3. *Все ограничения $\mathcal{M}$ принадлежат либо исключительно семейству Response, либо исключительно семейству Precedence.*

Доказательство. Докажем пункты по порядку.

1. Предположим, что в цикле присутствует запрещающее ограничение $C(a, b)$. Тогда трасса $\sigma = \langle a \rangle$ тривиально удовлетворяет этому ограничению, поскольку событие b в ней отсутствует. Остальные ограничения цикла не нарушаются, поскольку соответствующие события не активируются. Следовательно, $\sigma \models \mathcal{M}$, что противоречит несогласованности $\mathcal{M}$. Значит, все ограничения цикла обязаны быть обязывающими.
2. Предположим, что в $\mathcal{M}$ присутствует ограничение $\mathit{RespondedExistence}(a, b)$. Рассмотрим модель

   $$\mathcal{D}' = \mathcal{M} \cup \{\mathit{Existence}(b)\} \setminus \{\mathit{RespondedExistence}(a, b)\}.$$

   Поскольку замыкающее цикл ограничение удалено, $\mathcal{D}'$ не образует рассматриваемую структуру типа (II) без предцикла. Следовательно, существует трасса $\sigma \models \mathcal{D}'$. Из $\mathit{Existence}(b) \in \mathcal{D}'$ получаем $b \in \sigma$. Так как по циклу из b достигается a через обязывающие ограничения, имеем $a \in \sigma$. Тогда автоматически $\sigma \models \mathit{RespondedExistence}(a, b)$, откуда $\sigma \models \mathcal{M}$, что противоречит несогласованности $\mathcal{M}$.
3. Из пунктов 1 и 2 следует, что все ограничения модели $\mathcal{M}$ являются обязывающими, причём среди них нет ограничений семейства *RespondedExistence*. Следовательно, каждое ограничение принадлежит либо семейству *Response*, либо семейству *Precedence*.

Если все ограничения принадлежат семейству *Response*, то любая попытка построить конечную трассу приводит к бесконечному росту: появление одного события требует появления следующего, затем ещё одного и так далее, а замыкание цикла снова требует появления первого события. Следовательно, конечной удовлетворяющей трассы не существует.
Если все ограничения принадлежат семейству *Precedence*, то появление любого события цикла требует существования предшествующего события цикла раньше в трассе, что приводит к бесконечному регрессу к началу трассы. Следовательно, и в этом случае конечной удовлетворяющей трассы не существует.
Остаётся смешанный случай, когда в $\mathcal{M}$ одновременно имеются ограничение семейства *Response* и ограничение семейства *Precedence*. Пусть, например, это $Response(a, b)$ и $Precedence(c, b)$. Рассмотрим модель

$$\mathcal{D}'' = \mathcal{M} \cup \{Existence(c)\} \setminus \{Response(a, b),\ Precedence(c, b)\}.$$

Тогда существует трасса $\sigma \models \mathcal{D}''$. Из $Existence(c)$ получаем $c \in \sigma$, а по циклу обязывающих ограничений отсюда следует и $a \in \sigma$. Рассмотрим трассу $\sigma' = \langle \sigma, b \rangle$, полученную дописыванием события b в конец. Тогда $\sigma' \models Response(a, b)$, поскольку b идёт после a, и $\sigma' \models Precedence(c, b)$, поскольку c идёт до b. Все остальные ограничения сохраняются. Следовательно, $\sigma' \models \mathcal{M}$, что противоречит несогласованности $\mathcal{M}$.
Таким образом, смешанный случай невозможен.

□

Лемма 5 (Структура MIS типа (II) с предциклом)**.** *Пусть $\mathcal{M}$ — MIS типа (II), и граф следования $G(\mathcal{M})$ содержит предцикл*

$$x \to \cdots \to a$$

и цикл

$$a \to \cdots \to a.$$

Тогда в $\mathcal{M}$ не более одного запрещающего ограничения, и если оно существует, то его событием-целью является a.

Доказательство. Сначала предположим, что в $\mathcal{M}$ присутствуют два запрещающих ограничения $C(k, l)$ и $C(n, m)$. Без ограничения общности можно считать, что событие n встречается в предцикле раньше, чем k. Рассмотрим трассу

$$\sigma = \langle x, \ldots, n \rangle,$$

содержащую все события предцикла от x до n включительно. Все обязывающие ограничения, активируемые этими событиями, выполнены, а ограничение $C(n, m)$ выполнено, поскольку событие m в трассе отсутствует. Следовательно, $\sigma \models \mathcal{M}$, что противоречит несогласованности.

Теперь предположим, что запрещающее ограничение единственно и имеет вид $C(n, m)$, где $m \neq a$. Тогда та же трасса $\sigma = \langle x, \ldots, n \rangle$ снова удовлетворяет всем обязывающим ограничениям предцикла, а ограничение $C(n, m)$ выполняется, так как событие m в ней отсутствует. Получаем противоречие.

Следовательно, в $\mathcal{M}$ может быть не более одного запрещающего ограничения, и если оно есть, то его событием-целью обязательно является a. □

3.2. Алгоритм проверки кандидатов на MIS

Алгоритм поиска кандидатов возвращает семейство подмножеств ограничений, каждое из которых является лишь *потенциальным* MIS. Для каждого такого кандидата $\mathcal{M}$ необходимо ответить на вопрос: действительно ли $\mathcal{M}$ является минимальным несогласованным множеством.

Из теоремы 1 следует, что достаточно рассмотреть случаи типов (I)–(III). Для кандидатов типа (I) проверка опирается на фазу позиционной нормализации из предыдущего раздела. В этом случае алгоритм не только проверяет минимальность позиционного конфликта, но и, если конфликт отсутствует, нормализует модель: склеивает события, связанные обязательными directly-follows-требованиями, и заменяет исходный алфавит Σ укрупнённым алфавитом блоков $\widehat{\Sigma}$. Тем самым алгоритмы для типов (II) и (III) запускаются уже на модели без позиционных ограничений.

Центральная идея обоих алгоритмов носит конструктивный характер. Вместо проверки пустоты языка автоматными методами строится конкретная трасса-кандидат. Если такая трасса удовлетворяет $\mathcal{M}$, то $\mathcal{M}$ не является MIS. Если же при наиболее благоприятном выборе позиций событий построенная трасса всё равно нарушает одно критическое запрещающее ограничение, то это нарушение неустранимо, а значит, $\mathcal{M}$ является MIS.

3.2.1. Алгоритм для кандидатов типа (I)

Пусть кандидат $\mathcal{M}$ типа (I) состоит из позиционных ограничений и, возможно, непозиционных ограничений, инцидентных тем же событиям. Напомним, что в фазе позиционной нормализации из предыдущего раздела по позиционной части модели строится граф прямого следования

$$G_{df} = (V_{df}, E_{df}), \qquad V_{df} = \Sigma \cup \{\top\},$$

а также множество запрещённых дуг

$$F_{df} = \{(a, b) \mid NotChainSuccession(a, b) \in \mathcal{M}\}.$$

Если локальных конфликтов не обнаружено, то позиционная часть модели задаёт дизъюнктное объединение ориентированных цепочек и циклов. В этом случае позиционные ограничения не удаляются просто как «безвредные», а используются для *склейки* событий в блоки, чья взаимная позиция в любой допустимой трассе жёстко фиксирована. Именно эта нормализация должна быть выполнена до запуска алгоритмов 4 и 3.

Поэтому алгоритм для кандидатов типа (I) решает две задачи:

1. Проверяет, является ли рассматриваемое множество позиционных ограничений минимальным несогласованным.
2. Если нет, строит фактор-модель без позиционных ограничений. Такая модель получается склейкой directly-follows-блоков и заменой исходного алфавита на укрупнённый.

Для формализации введём вспомогательную процедуру нормализации.

Алгоритм 1. NormalizePositionPart($\mathcal{K}$)

Вход: модель $\mathcal{K}$ с выделенной позиционной частью.
Выход: либо Conflict, либо пара $(\widehat{\mathcal{K}}, \widehat{\Sigma})$, где $\widehat{\mathcal{K}}$ — модель без позиционных ограничений над укрупнённым алфавитом $\widehat{\Sigma}$.

1. Построить $G_{df}(\mathcal{K}) = (V_{df}, E_{df})$ и $F_{df}(\mathcal{K})$ по правилам позиционной нормализации.
2. Если существует вершина v с двумя различными входящими обязательными дугами, вернуть Conflict.
3. Если существует вершина v с двумя различными исходящими обязательными дугами, вернуть Conflict.
4. Если $E_{df}(\mathcal{K}) \cap F_{df}(\mathcal{K}) \neq \varnothing$, вернуть Conflict.
5. Если в $G_{df}(\mathcal{K})$ существует цикл, содержащий $\top$, вернуть Conflict.
6. Положить $\widehat{\mathcal{K}} \leftarrow \mathcal{K}$ и $\widehat{\Sigma} \leftarrow \Sigma$.
7. Пока $E_{df}(\widehat{\mathcal{K}}) \neq \varnothing$, выполнить следующие действия:
   (a) выбрать обязательную дугу $(u, v) \in E_{df}(\widehat{\mathcal{K}})$;

(b) создать новую вершину-блок $[uv]$ и положить $\widehat{\Sigma} \leftarrow (\widehat{\Sigma} \setminus \{u, v\}) \cup \{[uv]\}$;
(c) во всех ограничениях модели $\widehat{\mathcal{K}}$ заменить каждое вхождение u или v на $[uv]$;
(d) удалить из $\widehat{\mathcal{K}}$ позиционные ограничения, соответствующие дуге (u, v);
(e) перестроить $G_{df}(\widehat{\mathcal{K}})$ и $F_{df}(\widehat{\mathcal{K}})$ над алфавитом $\widehat{\Sigma}$: дуги вида (x, u) или (x, v) заменить на $(x, [uv])$, дуги вида (u, y) или (v, y) заменить на $([uv], y)$, а запрещённые дуги перенести аналогично;
(f) если после перестройки возник один из начальных конфликтов, вернуть Conflict.

8. Вернуть $(\widehat{\mathcal{K}}, \widehat{\Sigma})$.

Алгоритм 1 реализует в точности ту же схему, которая была описана в фазе позиционной нормализации из предыдущего раздела: если позиционная часть является согласованной, то все события, чья взаимная позиция жёстко фиксирована обязательными directly-follows-требованиями, последовательно склеиваются в единые блоки. В результате получается модель без позиционных ограничений, над укрупнённым алфавитом блоков.

На основе этой процедуры можно уже формально описать алгоритм проверки кандидата типа (I).

Алгоритм 2. VerifyMIS_I($\mathcal{M}$)

Вход: кандидат $\mathcal{M}$, содержащий позиционные ограничения.
Выход: либо MIS, либо Not-MIS и нормализованная модель $(\widehat{\mathcal{M}}, \widehat{\Sigma})$ без позиционных ограничений.

1. Вычислить $R \leftarrow \text{NormalizePositionPart}(\mathcal{M})$.
2. Если $R \neq$ Conflict, вернуть Not-MIS вместе с R.
3. Для каждого позиционного ограничения $C \in \mathcal{M}$:
   (a) вычислить $R_C \leftarrow \text{NormalizePositionPart}(\mathcal{M} \setminus \{C\})$;
   (b) если $R_C =$ Conflict, вернуть Not-MIS.
4. Вернуть MIS.

Теорема 3 (Корректность алгоритмов 1 и 2)**.** *Для кандидата $\mathcal{M}$ типа (I) алгоритм 2 возвращает* MIS *тогда и только тогда, когда $\mathcal{M}$ является минимальным несогласованным множеством типа (I). Если же он возвращает* Not-MIS*, то вместе с ответом строится нормализованная модель $(\widehat{\mathcal{M}}, \widehat{\Sigma})$ без позиционных ограничений, эквивалентная исходной модели по множеству допустимых поведений с точностью до склейки жёстко фиксированных блоков.*

Доказательство. Сначала докажем корректность алгоритма 1.

Если на начальных проверках обнаруживается конфликт, то по описанию фазы позиционной нормализации система обязательных directly-follows-требований является несогласованной: она либо нефункциональна, либо содержит одновременно обязательную и запрещённую дугу, либо принудительно замыкает цикл через $\top$. Следовательно, позиционная часть модели не допускает ни одной трассы, и возврат Conflict корректен.

Пусть теперь конфликтов нет. Тогда граф G_{df} представляет собой дизъюнктное объединение ориентированных цепочек и циклов, не содержащих принудительного конфликта. Каждая обязательная дуга (u, v) означает, что в любой трассе событие v жёстко привязано к событию u отношением непосредственного следования. Следовательно, замена пары (u, v) на один блок $[uv]$ не изменяет допустимое множество поведений, а лишь переходит к более крупной единице алфавита. Аналогично переносятся все инцидентные обязательные и запрещённые дуги, а также все непозиционные ограничения, в которых фигурировали u или v. Поскольку на каждом шаге число позиционных дуг уменьшается, процесс склейки завершается. В результате получается модель $(\widehat{\mathcal{K}}, \widehat{\Sigma})$ без позиционных ограничений, эквивалентная исходной по смыслу фазы 1. Значит, алгоритм 1 корректен.

Теперь перейдём к алгоритму 2.

Если алгоритм сразу возвращает Not-MIS, то $\text{NormalizePositionPart}(\mathcal{M})$ не обнаружил конфликта. Следовательно, сама модель $\mathcal{M}$ не является несогласованной моделью типа (I). Более того, алгоритм

одновременно строит её нормализованную версию $(\widehat{\mathcal{M}}, \widehat{\Sigma})$ без позиционных ограничений. Значит, $\mathcal{M}$ не является MIS, а полученная фактор-модель корректно передаётся на вход алгоритмам 4 и 3.

Если алгоритм возвращает Not-MIS *при проверке собственных подмножеств,* то существует позиционное ограничение $C \in \mathcal{M}$ такое, что

$$\text{NormalizePositionPart}(\mathcal{M} \setminus \{C\}) = \text{Conflict}.$$

Это означает, что уже собственное подмножество $\mathcal{M} \setminus \{C\}$ остаётся несогласованным на фазе позиционной проверки. Следовательно, $\mathcal{M}$ не минимально по включению среди несогласованных множеств и потому не является MIS.

Пусть алгоритм возвращает MIS. Тогда, во-первых, сама модель $\mathcal{M}$ даёт конфликт при нормализации, а значит, является несогласованной. Во-вторых, для каждого позиционного ограничения $C \in \mathcal{M}$ модель $\mathcal{M} \setminus \{C\}$ успешно проходит нормализацию, то есть после удаления любого одного ограничения позиционный конфликт исчезает. Следовательно, каждое собственное одноэлементное удаление согласовано, а значит, по определению MIS модель $\mathcal{M}$ является минимальным несогласованным множеством.

Итак, алгоритм 2 корректно различает случаи MIS и Not-MIS, а в случае Not-MIS одновременно выполняет ту самую склейку directly-follows-вершин и обновление алфавита, которые нужны для дальнейшего запуска алгоритмов типов (II) и (III). □

3.2.2. Алгоритм для кандидатов типа (III)

Пусть далее $\mathcal{M}$ обозначает модель, полученную после применения алгоритма 2, то есть модель без позиционных ограничений над укрупнённым алфавитом блоков. Пусть кандидат $\mathcal{M}$ задан двумя путями

$$p_1 : a \ \to b \to \cdots \to y \to x, \qquad p_2 : a \ \to c \to \cdots \to z \to x.$$

По лемме 2 запрещающее ограничение может стоять только в позиции $C(y, x)$ или $C(z, x)$, а по лемме 3 ровно одно из этих двух ограничений запрещающее. Без ограничения общности будем считать, что запрещающим является $C(z, x)$.

Ключевое наблюдение здесь таково. Чтобы проверить, может ли запрещающее ограничение $C(z, x)$ быть выполнено одновременно со всеми остальными ограничениями, необходимо построить трассу, в которой событие z расположено как можно раньше, а событие x — как можно позже. Именно поэтому путь p_1 обрабатывается «справа налево» с точки зрения влияния на позицию x, а путь p_2 — «слева направо» с точки зрения влияния на позицию z.

Алгоритм 3. VerifyMIS_III($\mathcal{M}, p_1, p_2$)

Вход: кандидат $\mathcal{M}$ типа (III), пути $p_1 : a \to \cdots \to y \to x$ и $p_2 : a \to \cdots \to z \to x$.
Выход: MIS или Not-MIS.

1. Если оба ограничения $C(y, x)$ и $C(z, x)$ запрещающие, вернуть Not-MIS.
2. Если оба ограничения $C(y, x)$ и $C(z, x)$ обязывающие, вернуть Not-MIS.
3. Если $C(y, x)$ запрещающее, поменять пути p_1 и p_2 местами.
4. Положить $\sigma \leftarrow \langle a \rangle$.
5. Для каждого ограничения $C_i(u, v) \in p_1$ в порядке от a до y:
   (a) если v уже присутствует в σ, перейти к следующему ограничению;
   (b) если C_i имеет тип *Response* или *Succession*, дописать v в конец σ;
   (c) если C_i имеет тип *Precedence*, вставить v непосредственно перед u в σ.
6. Добавить x в σ на максимально возможную позицию.
7. Для каждого ограничения $C_i(u, v) \in p_2$ в порядке от c до z:
   (a) если v уже присутствует в σ, перейти к следующему ограничению;

(b) если C_i имеет тип *Response* или *Succession*, вставить v непосредственно после u в σ;
(c) если C_i имеет тип *Precedence*, вставить v в начало σ.
8. Если $C(z, x) = \mathit{NotCoExistence}(z, x)$, вернуть MIS.
9. Если $\sigma \models C(z, x)$, вернуть Not-MIS; иначе вернуть MIS.

Теорема 4 (Корректность алгоритма 3)**.** *Алгоритм* VerifyMIS_III *возвращает* MIS *тогда и только тогда, когда* $\mathcal{M}$ *является минимальным несогласованным множеством.*

Доказательство. Если на начальных шагах алгоритм обнаруживает, что оба концевых ограничения либо запрещающие, либо обязывающие, то корректность ответа Not-MIS непосредственно следует из леммы 3: в обоих случаях явная удовлетворяющая трасса строится описанным в её доказательстве образом.

Пусть теперь одно из концевых ограничений обязывающее, а другое запрещающее, и без ограничения общности запрещающим является $C(z, x)$.

Если $C(z, x) = \mathit{NotCoExistence}(z, x)$, то событие x обязано присутствовать в трассе из-за ограничения $C(y, x)$, а событие z обязано присутствовать из-за внутренней структуры пути p_2. Следовательно, ни одна трасса не может удовлетворять $\mathcal{M}$, и алгоритм корректно возвращает MIS.

Остаётся случай, когда $C(z, x) \in \{\mathit{NotSuccession}, \mathit{NotChainSuccession}\}$. Если построенная трасса σ удовлетворяет $C(z, x)$, то по построению она удовлетворяет и всем остальным ограничениям обоих путей: каждое обязывающее ограничение обрабатывается вставкой своего события-цели в позицию, совместимую с его семантикой. Следовательно, $\sigma \models \mathcal{M}$, и ответ Not-MIS корректен.

Пусть теперь $\sigma \not\models C(z, x)$. Докажем, что это нарушение неустранимо. Алгоритм строит σ так, что событие x располагается как можно позже среди всех трасс, удовлетворяющих обязательствам пути p_1, а событие z — как можно раньше среди всех трасс, удовлетворяющих обязательствам пути p_2. Следовательно, любая другая трасса, удовлетворяющая всем обязывающим ограничениям обоих путей, может только сдвинуть x левее или z правее, но не наоборот. Поэтому если уже в наиболее благоприятной конфигурации событие x оказывается запрещённо расположенным относительно z, то в любой другой допустимой конфигурации это нарушение также сохранится. Значит, удовлетворяющей трассы не существует, то есть $L(\mathcal{M}) = \varnothing$.

Наконец, минимальность кандидата обеспечивается его структурой: удаление любого одного ограничения разрушает либо один из путей, либо критическое запрещающее ограничение, после чего удовлетворяющая трасса строится непосредственно. Следовательно, $\mathcal{M}$ действительно является MIS. □

3.2.3. Алгоритм для кандидатов типа (II)

Пусть далее $\mathcal{M}$ обозначает модель, полученную после применения алгоритма 2, то есть модель без позиционных ограничений над укрупнённым алфавитом блоков. Пусть кандидат $\mathcal{M}$ типа (II) содержит предцикл

$$x \to \cdots \to a$$

(возможно, пустой) и цикл

$$a \to \cdots \to a.$$

По леммам 4 и 5 возможны два принципиально различных сценария.

Во-первых, все ограничения могут быть обязывающими, причём тогда они обязаны принадлежать либо только семейству *Response*, либо только семейству *Precedence*. В этом случае несогласованность следует уже из самой циклической структуры.

Во-вторых, в кандидате может присутствовать ровно одно запрещающее ограничение $C(n, a)$, событие-цель которого совпадает с точкой входа в цикл. Именно этот случай требует явной конструктивной проверки.

Ключевое отличие от типа (III) состоит в том, что цикл допускает потенциальное повторное навязывание уже встречавшихся событий. Поэтому алгоритм запоминает множество уже добавленных событий и завершает обход циклической части в тот момент, когда очередное обязательство требует события, уже присутствующего в трассе.

Алгоритм 4. VerifyMIS_II($\mathcal{M}$, предцикл, цикл)

Вход: кандидат $\mathcal{M}$ типа (II) с предциклом $x \to \cdots \to a$ и циклом $a \to \cdots \to a$.
Выход: MIS или Not-MIS.

1. Если число запрещающих ограничений больше одного, вернуть Not-MIS.
2. Если существует запрещающее ограничение $C(n, m)$ и $m \neq a$, вернуть Not-MIS.
3. Если запрещающих ограничений нет:
   (a) если все ограничения имеют Response-тип, вернуть MIS;
   (b) если все ограничения имеют Precedence-тип, вернуть MIS;
   (c) если есть ограничение *RespondedExistence*, вернуть Not-MIS;
   (d) иначе вернуть Not-MIS.
4. Положить $\sigma \leftarrow \langle x \rangle$ и $\mathit{Seen} \leftarrow \{x\}$.
5. Для каждого ограничения $C_i(u, v)$ в предцикле:
   (a) если $v \in \mathit{Seen}$, перейти к следующему ограничению;
   (b) если C_i имеет тип *Response* или *Succession*, дописать v в конец σ;
   (c) если C_i имеет тип *Precedence*, вставить v непосредственно перед u;
   (d) обновить $\mathit{Seen} \leftarrow \mathit{Seen} \cup \{v\}$.
6. Для каждого ограничения $C_i(u, v)$ в цикле:
   (a) если $v \in \mathit{Seen}$ и $\sigma \not\models C_i$, вернуть MIS;
   (b) если $v \in \mathit{Seen}$ и $\sigma \models C_i$, завершить обход цикла;
   (c) если C_i имеет тип *Response* или *Succession*, вставить v непосредственно после u;
   (d) если C_i имеет тип *Precedence*, вставить v в начало σ;
   (e) обновить $\mathit{Seen} \leftarrow \mathit{Seen} \cup \{v\}$.
7. Если $\sigma \models C(n, a)$, вернуть Not-MIS; иначе вернуть MIS.

Теорема 5 (Корректность алгоритма 4)**.** *Алгоритм* VerifyMIS_II *возвращает* MIS *тогда и только тогда, когда* $\mathcal{M}$ *является минимальным несогласованным множеством.*

Доказательство. Если алгоритм обнаруживает более одного запрещающего ограничения или единственное запрещающее ограничение с событием-целью, отличным от a, то ответ Not-MIS корректен по лемме 5.

Если запрещающих ограничений нет, то по лемме 4 возможны лишь четыре случая. Если все ограничения принадлежат семейству *Response*, то цикл обязательств порождает бесконечный рост вправо, и конечной удовлетворяющей трассы не существует. Если все ограничения принадлежат семейству *Precedence*, то цикл порождает бесконечный регресс влево, и конечной удовлетворяющей трассы также не существует. В обоих случаях алгоритм корректно возвращает MIS. Если же в кандидате есть *RespondedExistence* или имеет место смешанный случай (*Response* вместе с *Precedence*), то та же лемма 4 гарантирует существование удовлетворяющей трассы, и ответ Not-MIS корректен.

Остаётся случай, когда существует ровно одно запрещающее ограничение $C(n, a)$. Тогда алгоритм строит трассу так, чтобы событие n было расположено максимально поздно вдоль предцикла, а событие a в циклической части — максимально рано. Если в этой экстремальной конфигурации ограничение $C(n, a)$ всё же нарушается, то никакая другая трасса, удовлетворяющая всем обязывающим ограничениям, уже не сможет исправить это нарушение: n нельзя сдвинуть ещё правее, а a нельзя сдвинуть ещё левее. Следовательно, удовлетворяющей трассы не существует, и ответ MIS корректен.

Если же построенная трасса удовлетворяет $C(n, a)$, то она удовлетворяет и всем прочим ограничениям предцикла и цикла по построению. Тогда $\sigma \models \mathcal{M}$, и алгоритм корректно возвращает Not-MIS.

Минимальность в случае ответа MIS следует из структуры кандидата: удаление любого одного ограничения разрушает либо замкнутый цикл, либо предцикл, либо критическое запрещающее ограничение. После этого удовлетворяющая трасса строится непосредственно. Следовательно, $\mathcal{M}$ является MIS. □

Таким образом, алгоритмы 2, 3 и 4 совместно с трёхфазным алгоритмом поиска кандидатов из предыдущего раздела образуют полный процедурный метод проверки несогласованности декларативной модели. Каждый из алгоритмов проверки выполняется за время, полиномиальное от числа ограничений кандидата: основные операции сводятся к линейному проходу по путям или циклам и элементарным операциям над трассой.

4. Экспериментальная оценка

4.1. Постановка экспериментов

В рамках работы был реализован прототип предложенного подхода к анализу несогласованности на языке Python. Реализация покрывает основные этапы метода: представление извлечённых Declare-ограничений, построение вспомогательных графовых структур, а также обнаружение и анализ несогласованностей. Графовые операции, включая построение directly-follows отношений, графов следования, компонент сильной связности и проверки, основанные на путях, были реализованы с использованием networkx, а агрегация и экспорт результатов — с помощью pandas.

В качестве алгоритма извлечения ограничений использовался MINERful [6]. Этот выбор был обусловлен тремя причинами. Во-первых, MINERful является быстрым и широко используемым декларативным майнером, специально предназначенным для автоматического извлечения Declare-спецификаций из журналов событий. Во-вторых, он использовался в двух основных работах, с которыми проводится сравнение ниже, что делает экспериментальную постановку максимально близкой к литературе. В-третьих, этот инструмент широко применяется для синтеза декларативных моделей процессов, а его текущая реализация поддерживает журналы событий в стандартных форматах.

Общий пайплайн предложенного подхода устроен следующим образом. Сначала по журналу событий с помощью MINERful извлекается декларативная модель. Затем синтезированная модель анализируется предложенным методом, который разделяет потенциальные несогласованности на три структурных класса: позиционные, циклические и двухпутевые паттерны. Для каждого класса соответствующая ветвь процедуры обнаруживает подмножества-кандидаты и проверяет, являются ли они минимальными несогласованными подмножествами. В результате формируются итоговые результаты анализа несогласованности.

Для разрешения несогласованности использовалась стандартная итеративная стратегия на основе подсчета culpability [8]. На каждой итерации вычислялось текущее множество минимальных несогласованных подмножеств, каждому ограничению назначалось значение culpability, равное числу MIS, в которые оно входит, после чего одно ограничение удалялось. При равных значениях culpability приоритет отдавался ограничениям с меньшими значениями support и confidence. В журналах выполнения нашей реализации при этом наблюдался устойчивый паттерн: удаление запрещающего ограничения часто значительно сильнее уменьшало число оставшихся MIS, чем удаление обязывающего ограничения. Это указывает на то, что на практике главным узким местом всего пайплайна исправления несогласованности является не сам этап обнаружения MIS, а выбор следующего ограничения для удаления.

Table 2. Real-life event logs used in the experimental evaluation

Таблица 2. Реальные журналы событий, использованные в экспериментальной оценке

Журнал событий	Соответствующий реальный процесс
BPI2012	Процесс оформления персональных кредитов и овердрафтов в нидерландской финансовой организации.
BPI2017	Онлайн-процесс подачи кредитной заявки в нидерландской финансовой организации; в этой версии процесса информационная система допускает несколько офферов в рамках одной заявки.
BPI2020: Domestic Declarations	Внутренние декларации командировочных расходов из системы управления командировками одного из нидерландских университетов.
BPI2020: International Declarations	Международные декларации командировочных расходов из той же университетской системы управления командировками.
BPI2020: Prepaid Travel Cost	Запросы на предварительную оплату командировочных расходов из той же системы.
BPI2020: Permit Log	Процесс согласования командировок, включая связанные события предварительной оплаты командировочных расходов и деклараций расходов, из системы управления командировками одного из нидерландских университетов.
BPI2020: Request for Payment	Запросы на оплату, формально не связанные с поездками, например представительские расходы или закупка оборудования для работы, зафиксированные в той же университетской административной системе.
Sepsis Cases	Траектории лечения пациентов с сепсисом, записанные в больничной ERP-системе.
Road Traffic Fine Management Process	Реальный журнал информационной системы, управляющей дорожными штрафами.

Использованные журналы событий перечислены в таблице 2. Они были выбраны по двум причинам. Во-первых, все они являются реальными журналами событий, а не синтетическими наборами данных. Во-вторых, часть из них уже использовалась в работах по синтезу и выявлению несогласованностей в декларативных моделях процессов, что позволяет провести прямое или близкое к прямому сравнение с ранее опубликованными результатами. Выбранные журналы покрывают несколько предметных областей: финансовые сервисы, университетские процессы согласования поездок и возмещения расходов, здравоохранение и публичное администрирование.

Семейство BPI 2020 оказалось особенно полезным, поскольку объединяет несколько родственных, но не идентичных процессов внутри одной административной среды. Это позволяет проверить устойчивость подхода на журналах, имеющих в целом похожую структуру согласования, но различающихся по размеру, гранулярности событий и семантике кейсов. Журналы Sepsis Cases и Road Traffic Fine Management Process были включены, чтобы расширить экспериментальную оценку за пределы финансовых и административных сценариев.

Для широкой оценки ниже приводятся результаты при support = 0.75, поскольку это значение support чаще всего используется в сопоставимой литературе. Дополнительно, для сравнения с culpability-подходом на журналах событий из BPI2017 [8], мы также приводим результаты для support = 0.85 и support = 0.95.

4.2. Результаты экспериментальной оценки

В таблице 3 приведены результаты нашей реализации на всех выбранных журналах при support = 0.75.

Из таблицы 3 следует несколько наблюдений. Во-первых, предложенный подход достаточно хорошо масштабируется на разнородном наборе реальных журналов событий. Наименьшее время работы было получено для Road Traffic Fine Management Process, где не было найдено ни одного MIS, а синтезированная модель уже оказалась согласованной. На противоположном конце находится

Table 3. Results of the proposed algorithm for support = 0.75

Таблица 3. Результаты предложенного алгоритма при support = 0.75

Журнал событий	# Ограничений	# MIS	# Удалённых	Потеря, %	Время работы, мс
BPI2012	365	13	6	1.64	543.0
BPI2017	1385	37	55	3.97	11595.6
Sepsis Cases	805	23	29	3.60	4730.9
Road Traffic Fine MP	96	0	0	0.00	14.5
Domestic Declarations	693	15	30	4.33	5534.2
International Declarations	2379	18	23	0.97	48917.6
Permit Log	3383	58	74	2.19	556945.9
Prepaid Travel Cost	1382	34	31	2.24	27140.4
Request for Payment	792	22	26	3.28	6625.4

Table 4. Comparison with [7] on BPI2012 at support = 0.75 and with [8] on BPI2017

Таблица 4. Сравнение с [7] на BPI2012 при support = 0.75 и с [8] на BPI2017

Support	# Ограничений	Несогласованность	#Удален.	Потеря, %	Время работы, мс
Di Ciccio и др. [7], BPI2012					
0.75	306	2 конфликта	176	57.52	9171.0
Эта работа, BPI2012					
0.75	365	13 MIS (2 конфликтующих множества)	6	1.64	543.0
Corea и др. [8], BPI2017					
0.75	305	28954 MIS	5	1.63	101099.0
0.85	232	731 MIS	1	0.43	9148.0
0.95	207	639 MIS	1	0.48	4695.0
Эта работа, BPI2017					
0.75	1385	37 MIS	55	3.97	11595.6
0.85	1350	37 MIS	55	4.07	10884.5
0.95	1223	35 MIS	59	4.82	7652.1

Permit Log: этот журнал дал наиболее крупную модель и наибольшее число MIS, что привело к максимальному времени работы. Это согласуется с тем, что Permit Log является самым крупным и структурно наиболее насыщенным представителем семейства BPI 2020.

Во-вторых, информационные потери остаются сравнительно небольшими для всех несогласованных журналов событий. Даже для наиболее сложных случаев, таких как Domestic Declarations и BPI2017, они не превышают 5 %. Это говорит о том, что процедура исправления удаляет только ограниченную долю извлечённого декларативного знания. В частности, International Declarations сочетает крупную исходную модель с очень низкими информационными потерями (0.97 %), что показывает: даже относительно плотные модели могут быть исправлены с умеренной потерей поведенческой информации.

В-третьих, результаты подтверждают, что размер модели сам по себе не полностью определяет сложность анализа. Например, International Declarations содержит значительно больше ограничений, чем BPI2017, но число MIS в нём меньше, а относительные информационные потери существенно ниже. Следовательно, решающую роль в практической производительности играет внутренняя структура графа зависимостей, а не только число извлечённых ограничений.

Чтобы поместить эти результаты в контекст, далее мы сравниваем их с двумя более ранними подходами из литературы.

В таблице 4 сначала наши результаты сопоставляются с экспериментом Di Ciccio и др. на BPI2012, основанным на MINERful. Это сравнение не является полностью прямым, поскольку их подход одновременно рассматривает несогласованность и избыточность, тогда как наша текущая реализация сосредоточена на обнаружении и устранении несогласованности через MIS. Тем не менее, такое сопоставление остаётся информативным.

Качественно обе работы показывают, что модель содержит лишь небольшое число регионов с высокой концентрацией конфликтов. Количественно наша модель содержит несколько больше начальных ограничений (365 против 306), однако время работы нашей реализации существенно меньше: 543.0 мс против 9171 мс, то есть, примерно в 16.9 раза меньше. При этом метрики не являются непосредственно взаимозаменяемыми: в более ранней работе сообщается число обнаруженных конфликтов и избыточностей, тогда как мы сообщаем число MIS и число удалённых ограничений. Поэтому наиболее осторожный вывод состоит в том, что на близкой, хотя и не идентичной конфигурации, предложенная нами реализация достигает существенно меньшего времени работы, оставаясь в том же качественном режиме анализа несогласованности.

Более прямое сравнение возможно с подходом Corea и др. на основе culpability, поскольку и их работа, и наша сообщают данные о количестве несогласованных множеств для моделей, извлечённых с помощью MINERful. В таблице 4 эти подходы сравниваются на BPI2017 для значений support 0.75, 0.85 и 0.95.

Таблица показывает, что модели в нашей постановке существенно крупнее при всех значениях support. При support = 0.75 наш пайплайн порождает 1385 ограничений против 305 в сравниваемой работе; при 0.85 и 0.95 различие также остаётся значительным. Это подтверждает, что точное воспроизведение более ранней экспериментальной постановки оказалось невозможным, вероятнее всего из-за различий в версии MINERful, параметрах постобработки и формате вывода.

Несмотря на это расхождение в размере моделей, сравнение времени работы остаётся интересным. При support = 0.75 наша реализация существенно быстрее: 11595.6 мс против 101099 мс, то есть примерно в 8.7 раза. При support = 0.85 и support = 0.95 подход на основе culpability быстрее в абсолютных значениях, однако он работает с существенно меньшими моделями. Поэтому более информативная интерпретация состоит не в том, что один метод равномерно доминирует над другим, а в том, что предложенный нами способ остаётся конкурентоспособным даже тогда, когда входная модель значительно плотнее.

Также заслуживает внимания расхождение в числе MIS. Corea и др. сообщают об очень большом числе начальных MIS при support = 0.75, тогда как соответствующая модель, полученная по нашему подходу, содержит только 37 MIS. Поскольку сами модели сильно различаются по размеру, эти числа не следует интерпретировать как прямое алгоритмическое преимущество одного метода над другим. Скорее, они подчёркивают чувствительность последующего анализа несогласованности к точной конфигурации. В этом смысле сравнение дополнительно усиливает мотивацию нашего структурного подхода: если настройки синтеза моделей резко меняют размер и плотность модели, то метод проверки несогласованности, работающий непосредственно с графовой структурой извлеченных декларативных ограничений, становится особенно привлекательным.

4.3. Обсуждение результатов

Экспериментальная оценка показывает, что предложенный подход практически применим на разнообразной коллекции реальных журналов событий. На журналах событий из банковской сферы, университетского администрирования, здравоохранения и публичного администрирования алгоритм позволил обнаруживать и устранять несогласованность, сохраняя информационные потери сравнительно низкими. Широкая оценка при support = 0.75 также показывает, что метод остаётся применимым к моделям с несколькими тысячами извлечённых ограничений.

Сравнение с предыдущими работами приводит к двум основным выводам. Во-первых, по сравнению с автоматным подходом Di Ciccio и др. предложенный нами метод достигает существенно меньшего времени работы в наиболее близкой сравнимой постановке, хотя две работы оптимизируют несколько разные целевые метрики. Во-вторых, по сравнению с подходом Corea и др. на основе culpability предложенная нами реализация остаётся конкурентоспособной даже при применении

к существенно более крупным моделям. Это показывает, что структурное ограничение на рассматриваемый фрагмент Declare действительно даёт практические вычислительные преимущества.

В то же время эксперименты выявляют и текущее ограничение реализации. Узким местом пайплайна исправления оказывается не обнаружение MIS как таковое, а стратегия выбора ограничения, которое следует удалить следующим. В частности, журналы выполнения показывают, что запрещающие ограничения часто оказывают значительно более сильное влияние на оставшуюся структуру несогласованности, чем обязывающие. Поэтому естественное направление дальнейшей работы состоит в том, чтобы заменить текущую чистую стратегию на основе culpability более информированной эвристикой, учитывающей также тип ограничения, его структурную роль в свидетеле и ожидаемое влияние на оставшийся граф несогласованности.

5. Обзор литературы

Исследования, посвящённые декларативному моделированию и анализу процессов, охватывают несколько тесно связанных направлений. Со стороны моделирования основы языка Declare и его роль в гибком моделировании процессов хорошо изучены [3, 4, 12]. Со стороны автоматического извлечения моделей важное направление связано с тем, как декларативные модели процессов могут быть получены из журналов событий в понятном и вычислительно эффективном виде, прежде всего с помощью Declare Maps Miner и MINERful [5, 6]. Эти подходы строят модели, которые затем становятся объектом дальнейшего анализа, включая проверку согласованности, понятности и избыточности.

Второе направление работ сосредоточено на операционном анализе декларативных моделей. К нему относятся runtime monitoring и conformance checking. Maggi и др. показали, как декларативные модели процессов, основанные на LTL, могут проверяться во время исполнения с использованием интерпретации на конечных трассах для обнаружения нарушений и конфликтующих ограничений [13]. De Leoni и др. предложили подход на основе выравниваний к проверке соответствия между журналами событий и декларативными моделями процессов [14]. Эти работы не направлены непосредственно на поиск несогласованности внутри самой модели, однако они тесно связаны с рассматриваемой задачей, поскольку изучают семантические последствия взаимодействия ограничений Declare на конечных трассах.

Третье важное направление связано с более выразительными декларативными языками, выходящими за пределы обычных ограничений потока управления. В частности, ряд работ расширяет Declare ограничениями, учитывающими данные, или многоперспективными конструкциями. Montali и др. ввели ограничения в Declare с учётом данных [15], а Borrego и Barba рассмотрели проверку соответствия и диагностику для декларативных моделей с данными [16]. Burattin и др. развили проверку соответствия для MP-Declare, многоперспективного расширения Declare [17]. Связанные работы по мониторингу бизнес-метаограничений над LTL_f и LDL_f также показывают, что темпоральные логики на конечных трассах образуют естественную основу для рассуждений о более выразительных декларативных спецификациях [18]. Эти результаты расширяют практическую область применения анализа декларативных процессов, но одновременно усиливают потребность в эффективных процедурах проверки согласованности.

Четвёртое направление касается инструментальной поддержки и практического применения. RuM объединяет в одном инструменте функции извлечения моделей, проверки соответствия, генерации журналов, мониторинга и редактирования моделей Declare и MP-Declare [19]. Более поздний инструмент Declare4Py предоставляет Python-библиотеку для декларативного process mining, включая проверку соответствия и извлечение моделей, а также упрощает проведение крупномасштабных экспериментов в Python-ориентированных средах [20]. Эти инструменты важны с инженерной точки зрения, поскольку делают рабочие процессы декларативного анализа воспроизводимыми и расширяемыми.

Ближе всего к нашей постановке находятся работы, непосредственно посвящённые несогласованности и избыточности в декларативных моделях процессов. Di Ciccio и др. предложили автоматный подход к устранению несогласованностей и избыточностей с помощью произведений автоматов и проверок включения языков [7]. Corea и др. предложили подход, основанный на мере culpability (виновности), в центре которого находятся минимальные несогласованные подмножества и измерение несогласованности [8, 10]. Kuhlmann и др. исследовали применение программирования набора ответов (ASP) для декларативных моделей процессов, избегая явного построения автоматов, но сохраняя логический подход к проверке выполнимости и связанным задачам [21]. Наконец, Schützenmeier и др. изучили автоматное семантическое сравнение моделей Declare, что дополнительно показывает сохраняющуюся важность автоматных методов в этой области [22].

Наша работа наиболее близка по духу к перечисленным подходам к анализу несогласованности, но отличается от них в двух отношениях. Во-первых, мы рассматриваем ограниченный фрагмент Declare, для которого несогласованность может быть охарактеризована структурно через графовые свидетельства. Во-вторых, вместо произведений автоматов, планирования или универсальных логических кодировок мы используем графовый поиск кандидатов с последующей специализированной верификацией. В этом смысле наша работа занимает промежуточное положение между выразительными, но вычислительно тяжёлыми общими подходами к рассуждению и эвристическими методами на основе построения минимальных несогласованных множеств, которые не дают полной формализации обнаруживаемых ими структур.

Заключение

В данной работе была рассмотрена задача анализа несогласованности декларативных моделей процессов для ограниченного фрагмента языка Declare. Мы предложили графовый взгляд на эту задачу: зависимости, индуцируемые ограничениями, представляются в виде графа следования задач, а несогласованность характеризуется с помощью трёх типов структурных свидетелей: позиционных, циклических и двухпутевых. На основе этой характеризации была построена трёхфазная процедура обнаружения кандидатных несогласованных подмножеств, после чего была уточнена внутренняя структура минимальных несогласованных подмножеств каждого типа. Эти структурные результаты были использованы для вывода специализированных процедур верификации, позволяющих проверить, является ли найденный кандидат минимальным несогласованным подмножеством. В отличие от автоматных подходов, предложенный метод избегает явного построения автоматов и вместо этого опирается на графовые процедуры и конструктивные рассуждения о трассах. Мы также реализовали предложенный подход и провели его экспериментальную оценку на наборе реальных журналов событий из нескольких предметных областей. Результаты экспериментов показали, что метод практически применим и обеспечивает конкурентоспособное время работы при сравнительно небольших информационных потерях.

В то же время у данной работы есть ряд ограничений. Основное теоретическое ограничение состоит в том, что в общем случае граф может содержать экспоненциально много пар различных путей с общими начальной и конечной вершинами. Следовательно, хотя процедура обнаружения кандидатов имеет структурную мотивацию, семейство потенциальных кандидатов типа (III) всё ещё может быть экспоненциальным в худшем случае. Это означает, что текущая формулировка пока не устраняет полностью комбинаторный взрыв, присущий структурам, основанным на путях.

Естественным направлением дальнейшей работы является уточнение структурной характеристики MIS. Один из возможных путей состоит в усилении описания свидетелей типа (III), чтобы рассматривать меньше неминимальных кандидатных подмножеств. Другой путь состоит в формулировании правил канонизации, позволяющих выбирать репрезентативное потенциальное подмножество для заданной пары вершин или заданного паттерна свидетеля, вместо перебора всех

возможных комбинаций путей. Такие уточнения могут сохранить полноту метода и при этом существенно сократить практическое пространство поиска.

Второе направление дальнейшей работы связано не с обнаружением несогласованности, а с её исправлением. В текущей реализации выбор ограничения для удаления основан на culpability, однако экспериментальная оценка показывает, что такая стратегия не всегда оптимальна. В частности, журналы экспериментов показывают, что удаление запрещающего ограничения часто сильнее уменьшает оставшуюся структуру несогласованности, чем удаление обязывающего ограничения. Это мотивирует разработку более информированных стратегий исправления, которые учитывают не только число MIS, содержащих данное ограничение, но и его тип, его структурную роль в свидетеле и ожидаемый эффект на оставшийся граф противоречий.

Наконец, естественным продолжением данной работы является анализ избыточных декларативных моделей. После того как обнаружение и исправление несогласованности были формализованы в графовых терминах, естественно поставить вопрос о том, могут ли сходные структурные идеи использоваться для характеризации избыточности, эффективного выявления избыточных ограничений и связи свидетелей избыточности с тем же графом следования задач. В этом смысле исследование избыточных моделей представляется прямым и перспективным расширением подхода, разработанного в данной работе.

References

[1] W. M. P. van der Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer, 2011, 352 pp.

[2] W. M. P. van der Aalst, M. L. Rosa, and F. M. Santoro, “Business process management”, *Business & Information Systems Engineering*, vol. 58, no. 1, pp. 1–6, 2016.

[3] M. Pesic, H. Schonenberg, and W. M. P. van der Aalst, “Declare: Full support for loosely-structured processes”, in *Proceedings of the 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC)*, 2007, pp. 287–300.

[4] W. M. P. van der Aalst, M. Pesic, and H. Schonenberg, “Declarative workflows: Balancing between flexibility and support”, *Computer Science - Research and Development*, vol. 23, no. 2, pp. 99–113, 2009.

[5] F. M. Maggi, R. P. J. C. Bose, and W. M. P. van der Aalst, “Efficient discovery of understandable declarative process models from event logs”, in *Proceedings of the International Conference on Advanced Information Systems Engineering*, 2012, pp. 270–285.

[6] C. D. Ciccio and M. Mecella, “On the discovery of declarative control flows for artful processes”, *ACM Transactions on Management Information Systems*, vol. 5, no. 4, 24:1–24:37, 2015.

[7] C. D. Ciccio, F. M. Maggi, M. Montali, and J. Mendling, “Resolving inconsistencies and redundancies in declarative process models”, *Information Systems*, vol. 64, pp. 425–443, 2017.

[8] C. Corea, M. Deisen, and P. Delfmann, “Resolving inconsistencies in declarative process models based on culpability measurement”, in *Proceedings of the 14th International Conference on Wirtschaftsinformatik (WI)*, 2019, pp. 1–16.

[9] A. P. Sistla and E. M. Clarke, “The complexity of propositional linear temporal logics”, *Journal of the ACM*, vol. 32, no. 3, pp. 733–749, 1985.

[10] C. Corea, J. Grant, and M. Thimm, “Measuring inconsistency in declarative process specifications”, in *Proceedings of the International Conference on Business Process Management*, 2022, pp. 289–306.

[11] A. Hunter and S. Konieczny, “Measuring inconsistency through minimal inconsistent sets”, in *Proceedings of the Eleventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2008)*, 2008, pp. 358–366.

[12] C. D. Ciccio and M. Montali, "Declarative process specifications", in *Process Mining Handbook*, Springer, 2022, pp. 119–147.

[13] F. M. Maggi, M. Westergaard, M. Montali, and W. M. P. van der Aalst, "Runtime verification of LTL-based declarative process models", in *Runtime Verification*, ser. Lecture Notes in Computer Science, vol. 7186, 2012, pp. 131–146.

[14] M. de Leoni, F. M. Maggi, and W. M. P. van der Aalst, "Aligning event logs and declarative process models for conformance checking", in *Business Process Management*, ser. Lecture Notes in Computer Science, vol. 7481, Springer, 2012, pp. 82–97.

[15] M. Montali, F. Chesani, P. Mello, and F. M. Maggi, "Towards data-aware constraints in Declare", in *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, 2013, pp. 1391–1396.

[16] D. Borrego and I. Barba, "Conformance checking and diagnosis for declarative business process models in data-aware scenarios", *Expert Systems with Applications*, vol. 41, no. 11, pp. 5340–5352, 2014.

[17] A. Burattin, F. M. Maggi, and A. Sperduti, "Conformance checking based on multi-perspective declarative process models", *Expert Systems with Applications*, vol. 65, pp. 194–211, 2016.

[18] G. D. Giacomo, R. D. Masellis, M. Grasso, F. M. Maggi, and M. Montali, "Monitoring business metaconstraints based on LTL and LDL for finite traces", in *Business Process Management*, ser. Lecture Notes in Computer Science, vol. 8659, 2014, pp. 1–17.

[19] A. Alman, C. D. Ciccio, D. Haas, F. M. Maggi, and J. Mendling, "Rule mining in action: The RuM toolkit", in *Proceedings of the ICPM Doctoral Consortium and Tool Demonstration Track 2020*, ser. CEUR Workshop Proceedings, vol. 2703, 2020, pp. 51–54.

[20] I. Donadello, F. Riva, F. M. Maggi, and A. Shikhizada, "Declare4py: A python library for declarative process mining", in *BPM 2022 Demos & Resources Forum*, ser. CEUR Workshop Proceedings, vol. 3216, 2022, pp. 117–121.

[21] I. Kuhlmann, C. Corea, and J. Grant, "An ASP-based framework for solving problems related to declarative process specifications", in *Proceedings of the 21st International Workshop on Nonmonotonic Reasoning*, 2023, pp. 129–132.

[22] N. Schützenmeier, M. Käppel, L. Ackermann, S. Jablonski, and S. Petter, "Automaton-based comparison of Declare process models", *Software and Systems Modeling*, vol. 22, pp. 667–685, 2023.