DOI: 10.15514/ISPRAS-2026-38(4)-26

TDM: формальная модель данных для интеграции структурированных и неструктурированных данных

Д.Ю. Турдаков, ORCID: 0000-0001-8745-0984<turdakov@ispras.ru>
В.Д. Майоров, ORCID: 0000-0003-3882-1114<vmayorov@ispras.ru>
М.А. Рындин, ORCID: 0000-0002-7504-3975 <mxrynd@ispras.ru>

Институт системного программирования им. В.П. Иванникова РАН,
Россия, 109004, г. Москва, ул. А. Солженицына, д. 25.

Аннотация. Современные информационно-аналитические системы совместно обрабатывают документы, результаты извлечения информации, базы знаний и делают экспертные проверки. Документные форматы сохраняют исходный контент, графы знаний описывают семантические связи, но происхождение данных, статусы проверки и жизненный цикл извлечённых фактов обычно задаются внешними механизмами. Это затрудняет интеграцию источников, повторную обработку документов и сопровождение предметно-ориентированных алгоритмов. В статье предлагается TDM (Talisman Data Model) – формальная модель для совместного представления документов, извлечённых фактов, предметных областей и баз знаний. Документ и база знаний рассматриваются как контейнеры типизированных фактов с указанием происхождения, статуса проверки, структурного положения и предметного контекста. Модель единообразно описывает исходный контент, результаты извлечения, экспертную верификацию, графовую проекцию и перенос знаний между предметными областями. Основной результат работы – математическая модель корректных состояний TDM-контейнера и операций, сохраняющих эти состояния. Показано, что TDM может служить формальным интеграционным слоем для компонентов интеллектуальной информационно-аналитической системы; практическая применимость модели подтверждается её использованием в платформе «Talisman».

Ключевые слова: Talisman, модель данных; информационно-аналитическая система; документ; факт; база знаний; машинное обучение; верификация человеком; граф знаний; медиатор; генерация с дополненным поиском (RAG); графовая генерация с дополненным поиском (GraphRAG).

Для цитирования: Турдаков Д.Ю., Майоров В.Д., Рындин М.А. TDM: формальная модель данных для интеграции структурированных и неструктурированных данных. Труды ИСП РАН, 2026, том 38, вып. 4, часть 2, стр. 193–214. DOI: 10.15514/ISPRAS-2026-38(4)-26.

TDM: A Formal Data Model for Integrating Structured and Unstructured Data

D.Yu. Turdakov, ORCID: 0000-0001-8745-0984<turdakov@ispras.ru>
V.D. Mayorov, ORCID: 0000-0003-3882-1114<vmayorov@ispras.ru>
M.A. Ryndin, ORCID: 0000-0002-7504-3975<mxrynd@ispras.ru>

Ivannikov Institute for System Programming of the Russian Academy of Sciences,
25, Alexander Solzhenitsyn st., Moscow, 109004, Russia.

Abstract. Modern information and analytical systems jointly process documents, information extraction results, knowledge bases, and expert validation. Document formats preserve source content and knowledge graphs capture semantic links, but provenance, verification status, and the lifecycle of extracted facts are usually maintained by external mechanisms. This fragmentation complicates source integration, iterative document processing, and maintenance of domain-specific analytical algorithms. The paper introduces TDM (Talisman Data Model), a formal model for representing documents, extracted facts, domains, and knowledge bases within a single framework. In TDM, documents and knowledge bases are containers of typed facts associated with provenance, verification status, structural position, and domain context. The model uniformly describes source content, extraction results, expert validation, knowledge graph projection, and cross-domain knowledge transfer. The main result is a mathematical model of valid TDM-container states and state-preserving operations. TDM can serve as a formal integration layer for intelligent information and analytical systems; its practical applicability is demonstrated by its use in the Talisman integration platform.

Keywords: Talisman; data model; information-analytical system; document; fact; knowledge base; machine learning; human-in-the-loop; knowledge graph; retrieval-augmented generation (RAG); graph retrieval-augmented generation (GraphRAG).

For citation: Turdakov D.Yu., Mayorov V.D., Ryndin M.A. TDM: A Formal Data Model for Integrating Structured and Unstructured Data. Trudy ISP RAN/Proc. ISP RAS, 2026, vol. 38, issue 4, part 2, pp. 193-214 (in Russian). DOI: 10.15514/ISPRAS-2026-38(4)-26.

1. Введение

Современные информационно-аналитические системы объединяют данные из источников, различающихся не только форматом и структурой, но и используемой семантикой. Одни и те же объекты, свойства и отношения могут описываться в разных предметных областях с помощью различных типов, схем и систем идентификаторов. Поэтому интеграция данных требует механизмов, позволяющих устанавливать соответствия между такими представлениями, переносить информацию из одной предметной области в другую и контролировать сохранение её смысла при преобразовании.

Одновременно информационно-аналитические системы совместно обрабатывают неструктурированные документы, результаты автоматического извлечения информации и структурированные базы знаний. В таких системах необходимо не только сопоставлять связанные по смыслу представления, но и сохранять происхождение данных, связь извлечённых значений с фрагментами исходных документов, статус проверки и историю преобразований. Вероятностная природа современных модулей машинного обучения (ML) и больших языковых моделей (LLM) дополнительно требует поддержки экспертной верификации, учёта ошибок и, в некоторых случаях, повторной обработки данных [1].

Существующие средства обычно описывают отдельные части этого процесса. Документные форматы сохраняют исходное содержимое и структуру документов, графы знаний представляют нормализованные семантические связи, средства описания происхождения данных фиксируют источники и историю получения утверждений, а методы сопоставления онтологий и обмена данными задают соответствия между отдельными схемами. Однако совместный жизненный цикл исходного фрагмента, автоматически извлечённого утверждения, результата его проверки и соответствующего элемента базы знаний, включая

перенос между предметными областями, как правило, определяется внешней логикой прикладной системы.

Для устранения этой фрагментации требуется единая модель состояния, в которой исходный контент, извлечённые факты, их структурное положение, происхождение, идентификаторы и статусы проверки описываются согласованным образом. Такая модель должна задавать правила сопоставления и переноса информации между предметными областями с максимальным сохранением информации, позволять проверять структурную корректность состояния и фиксировать связь фактов с источниками.

В настоящей работе предлагается TDM (Talisman Data Model) — формальная модель данных для совместного представления документов, извлечённых фактов, предметных областей и баз знаний. TDM рассматривает документ и базу знаний как контейнеры типизированных фактов и формализует отображения, позволяющие выразить содержание исходной базы знаний в терминах целевой предметной области. Факт в этой модели является не только элементом семантического представления, но и объектом жизненного цикла обработки: первоначально он может быть сформирован автоматически и ожидать проверки, а затем — связан с источником, проверен, подтверждён, объединён с эквивалентным фактом, перенесён в другую предметную область либо удалён из контейнера или исключён из состава актуального знания без физического удаления.

Модель TDM была разработана при создании платформы «Talisman», предназначенной для построения информационно-аналитических систем, объединяющих сбор данных, обработку документов, машинное обучение, экспертную проверку, хранилища знаний и прикладные сервисы анализа [2]. При этом предлагаемый формализм не зависит от конкретной программной платформы и может использоваться как интеграционный слой в других документно-ориентированных конвейерах.

Основной вклад работы состоит в следующем:

1. предложена единая структура типизированного факта и TDM-контейнера, позволяющая в одном формализме представлять исходный контент, извлечённые утверждения, происхождение значений, статусы и идентификационные классы;
2. задана формальная модель корректных состояний контейнера: предикат структурной корректности, графовая проекция, покрытие графа деревьями атрибутов и приведённая нормальная форма базы знаний;
3. формализован перенос информации между предметными областями: проекции баз знаний, операционные правила отображения и медиаторы, для которых доказана достаточность полного отображения для построения полной проекции.

Статья организована следующим образом. В разделе 2 обсуждается связь TDM с существующими подходами. Раздел 3 вводит базовые определения факта, документа и базы знаний. В этом же разделе приведён сквозной пример применения модели. В разделе 4 формулируются ограничения целостности модели TDM. Раздел 5 посвящён графовой проекции, покрытию графа знаний и деревьям раскрытия. В разделе 6 описываются операции TDM и их назначение, в разделе 7 вводится приведённая нормальная форма. Центральный раздел 8 формализует отображения предметных областей и медиаторы. Раздел 9 обсуждает ограничения модели.

2. Связь с существующими подходами

TDM сопоставляется с близкими подходами по трём вопросам: что является базовым состоянием системы, как фиксируются происхождение и проверка утверждений, и как переносится информация между предметными областями.

Модель RDF, язык OWL и графы свойств хорошо описывают нормализованный семантический граф; семейство спецификаций PROV, RDF-star и язык SHACL –

происхождение, утверждения об утверждениях и структурные ограничения; ontology matching и data exchange – сопоставление схем и перенос данных [3-7]. TDM выбирает другой базовый уровень: граф знаний, происхождение, подтверждение, логическое удаление и междоменный перенос рассматриваются как части одного контейнера фактов.

Роль TDM – задать модель состояния для документно-ориентированного конвейера, совместимую с RDF/OWL, SHACL, PROV, языками запросов и протоколами агентной инфраструктуры. Эти технологии могут быть форматами реализации или проекциями TDM, но инварианты совместного жизненного цикла факта в них обычно задаются внешним кодом.

Протоколы Model Context Protocol (MCP) и Agent-to-Agent (A2A) решают инфраструктурную задачу: подключают инструменты, ресурсы и агентов. Они не определяют, что считать фактом, как хранить его источник, статус, идентификаторы, удаление и перенос. Модель данных TDM может использоваться как содержательная полезная нагрузка таких протоколов: результат инструмента или сообщение агента становится типизированным фактом или фрагментом контейнера.

В RAG- и GraphRAG-конвейерах фрагменты текста и автоматически построенные графы используются как контекст генерации [8-10], а агентные системы обмениваются результатами вызова инструментов [11-13]. Во всех этих случаях полезно иметь общий формат состояния, отделяющий исходный материал, извлечённые кандидаты, проверенные факты и связи между ними.

Основные различия между TDM и рассмотренными подходами с точки зрения представляемого состояния, происхождения информации, проверки ограничений и междоменного переноса обобщены в табл. 1.

Табл. 1. Позиционирование TDM относительно близких подходов.
Table 1. Positioning of TDM Relative to Related Approaches.

Подход	Что покрывает	Что задаёт TDM на базовом уровне
RDF/OWL, LPG	Семантический граф и запросы к нему	Контейнер фактов, из которого граф является проекцией
PROV, RDF-star	Происхождение и утверждения об утверждениях	Происхождение как часть жизненного цикла факта
SHACL, схемы БД	Проверка структурных ограничений	Единый предикат корректности контейнера и операций над ним
Ontology matching, data exchange	Сопоставление схем и перенос данных	Построение базы знаний, корректной в целевой предметной области
MCP (Model Context Protocol)	Протокол подключения инструментов, ресурсов и контекста к LLM- и агентным системам	Модель содержательного состояния: факты, происхождение, проверка и перенос между предметными областями
RAG/GraphRAG, агентные системы	Поиск контекста и обмен результатами работы компонентов	Проверяемое состояние с фрагментами, фактами, статусами и графовой проекцией

3. Базовая модель факта, документа и базы знаний

3.1 Факт и предметная область

TDM описывает состояние документа или базы знаний относительно предметной области. Предметная область задаёт словарь допустимых типов фактов и правила их интерпретации, а базовой единицей состояния является факт.

Для понимания дальнейших определений удобно использовать следующую фразу в качестве сквозного примера:

«Иван Петров родился в Москве 12.03.1980»

В TDM исходное предложение сохраняется в аннотации типа Content. Из него выделяются имя «Иван Петров», название города «Москва» и дата «12.03.1980», которые представляются аннотациями типов Name, CityName и Date соответственно. Для каждой извлечённой аннотации сегмент указывает позицию соответствующего фрагмента в исходном тексте. Концепт типа Person представляет Ивана Петрова и содержит его имя в качестве атрибута, а концепт типа City представляет Москву и содержит название города. Связь типа BornIn соединяет концепты человека и города и выражает утверждение о том, что Иван Петров родился в Москве, а также содержит дату рождения, как атрибут. Полное представление этого примера в виде TDM-контейнера приведено в табл. 2.

Табл. 2. Фрагмент TDM-контейнера для примера.
Table 2. Fragment of a TDM Container for the Example.

id	**вид**	**тип**	m	v
a_0	annotation	Content	document	«Иван Петров родился в Москве 12.03.1980.»
c_1	concept	Person	$\emptyset$	$\emptyset$
a_1	annotation	Name	c_1	«Иван Петров»
s_1	segment	range	a_1	$(a_0, [0,12])$
c_2	concept	City	$\emptyset$	$\emptyset$
a_2	annotation	CityName	c_2	«Москва»
s_2	segment	range	a_2	$(a_0, [22,28])$
r_1	relation	BornIn	$\emptyset$	$\emptyset$
a_3	annotation	personRole	r_1	c_1
a_4	annotation	cityRole	r_1	c_2
a_5	annotation	Date	r_1	12.03.1980
s_3	segment	range	a_5	$(a_0, [29,39])$

Определение 1 (Предметная область). Предметная область θ – это пара

$$\theta = (l_\theta, T_\theta),$$

где l_θ – уникальный идентификатор предметной области, а T_θ – множество типов фактов, определённых в этой области. В пределах одной предметной области метка типа однозначно определяет его сигнатуру.

Предметная область задаёт пространство имён типов, локальные правила корректности и контекст интерпретации. Разные документы и базы знаний могут относиться к разным областям, а перенос между ними задаётся проекциями и медиаторами (раздел 8).

Определение 2 (Тип факта). Тип факта $t \in T_\theta$ – это пара

$$t = (l, \Sigma),$$

где l – метка типа, уникальная в предметной области, а Σ – сигнатура типа. Сигнатура задаёт допустимые значения, атрибуты и структурные связи факта. В примере выше Name, Person, BornIn – метки типов фактов.

Определение 3 (Факт). Факт – это упорядоченная пятёрка

$$f = (id, t, M, s, v) \in F,$$

где id – непустое конечное множество идентификаторов факта, t – тип факта, M – структурная связь факта, s – статус факта, v – значение факта.

Далее компоненты факта обозначаются через id(f), t(f), m(f), s(f) и v(f). Единая форма позволяет применять к аннотациям, группам, концептам, связям и сегментам одни и те же операции жизненного цикла: добавление, обновление, подтверждение, логическое удаление, идентификационное объединение, разделение и перенос между предметными областями.

После создания факт проходит несколько стадий обработки. Первоначально он рассматривается как неподтверждённый результат извлечения или ввода данных. Затем факт может быть проверен и признан корректным, удален как ошибочный или исключён из состава актуального знания. Последовательность изменений состояния факта от его создания до подтверждения, отклонения или удаления называется жизненным циклом факта. Текущее состояние факта фиксируется его статусом.

Определение 4 (Статус факта). Статус факта фиксирует текущее состояние факта в его жизненном цикле:

$$s(f) \in \{\text{new,approved,deleted}\}.$$

Статус «новый» (new) означает, что факт создан, но ещё не подтверждён. Статус «подтверждён» (approved) присваивается после проверки человеком, алгоритмом или другим *актором*, имеющим право подтверждения. Статус «удалён» (deleted) используется для логического удаления: факт исключён из состава актуального знания; в зависимости от реализации он может сохраняться в истории обработки либо быть физически удалён из контейнера. Допустимые переходы между состояниями факта и соответствующие им операции приведены в табл. 3.

Табл. 3. Базовые переходы статуса факта.
Table 3. Basic Transitions Between Fact Statuses.

Переход	**Операция**	**Описание**
$\emptyset \rightarrow$ new	$\mathcal{O}^Z_{add}$	создан новый факт, ещё не прошедший проверку
new → approved	$\mathcal{O}^Z_{approve}$	факт подтверждён в фиксированном контексте проверки
new → deleted	$\mathcal{O}^Z_{delete}$	кандидат признан неверным или неприменимым
approved → deleted	$\mathcal{O}^Z_{logical-delete}$	подтверждённый ранее факт исключён из активного знания
deleted → new	$\mathcal{O}^Z_{update}$	восстановление выполняется созданием нового факта или версии

Определение 5 (Множество идентификаторов). Пусть ID – множество всех допустимых идентификаторов фактов. Каждому факту $f \in F$ сопоставляется непустое конечное множество идентификаторов

$$id(f) \in P_{\text{fin}}(\text{ID}) \setminus \{\emptyset\}.$$

Множество $id(f)$ представляет идентификационный класс эквивалентных описаний одного факта. Два факта называются идентификационно эквивалентными, если их множества идентификаторов пересекаются:

$$f_1 \sim_{id} f_2 \quad \Leftrightarrow \quad id(f_1) \cap id(f_2) \neq \emptyset.$$

3.2 Виды фактов

TDM вводит пять видов фактов: сегменты, аннотации, группы, концепты и связи. Соответствующие множества обозначаются F_s, F_a, F_g, F_c и F_r. Эти множества попарно не пересекаются: каждый факт контейнера имеет ровно один вид. Назначение и структурная роль каждого вида фактов представлены в табл. 4.

Табл. 4. Виды фактов TDM.
Table 4. Types of TDM Facts.

Вид	Описание	Структурная роль
Сегмент F_s	Фиксирует происхождение значения.	m(f_s)=target(f_s), где target(f_s) – целевая аннотация, для значения которой фиксируется происхождение; v(f_s)=(ref(f_s), span(f_s)), где ref(f_s) задаёт источник, а span(f_s) – фрагмент источника.
Аннотация F_a	Хранит собственное значение $v(a)$.	$m(a)$ указывает владельца: группу, концепт или связь; значение должно быть допустимо для типа аннотации.
Группа F_g	Задаёт составной атрибут.	Структурирует аннотации и вложенные группы; $m(g) \in F_g \cup F_c \cup F_r$.
Концепт F_c	Представляет объект, понятие или сущность предметной области.	Имеет атрибуты и может участвовать в связях.
Связь F_r	Представляет отношение между концептами.	Участники задаются ролевыми атрибутами, дополнительные свойства – обычными атрибутами связи.

Ролевым атрибутом связи называется атрибут, значение которого указывает на концепт-участник связи и тип которого соответствует одной из ролей, разрешённых сигнатурой типа связи. Обычные атрибуты описывают свойства самого факта-связи. Направление связи, если оно требуется в предметной области, задаётся семантикой ролей. Например, для бинарной связи могут использоваться роли source/target или part/whole; отдельный атрибут направления для этого не требуется.

Сегменты позволяют хранить происхождение значений внутри модели: извлечённая аннотация может быть связана с фрагментом исходного текста, изображением, таблицей или промежуточным значением.

Для факта $f \in F_g \cup F_c \cup F_r$ множество его непосредственных атрибутов обозначается

$$A(f) = \{x \in F_a \cup F_g \mid m(x) = f\}.$$

Транзитивное замыкание атрибутов обозначается $A^+(f)$. Оно строится только по отношению владения атрибутом и не включает концепты, на которые могут ссылаться значения аннотаций.

Определение 6 (Участники связи). Участниками связи $r \in F_r$ называется множество концептов, на которые указывают значения ролевых атрибутов этой связи:

$$\text{Participants}(r) = \{c \in F_c \mid \exists\, a \in A(r): a \text{ является ролевым атрибутом и } v(a) = c\}.$$

3.3 Документ и база знаний

Определение 7 (Документ). Документ $\mathfrak{d}$ предметной области θ – это контейнер фактов

$$\mathfrak{d} = (id_{\mathfrak{d}}, F_{\mathfrak{d}}, \theta),$$

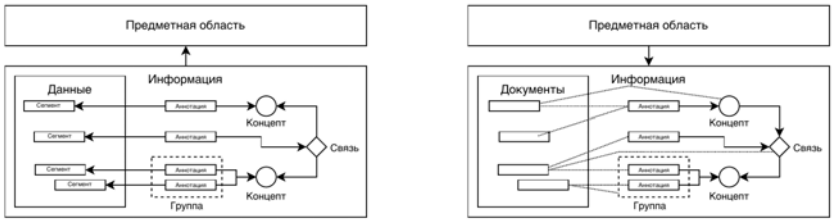
где $id_{\mathfrak{d}}$ – непустое конечное множество идентификаторов документа, $F_{\mathfrak{d}}$ – конечное множество фактов документа, а θ – его предметная область.

Неструктурированное содержимое документа хранится в значениях аннотаций. Метаданные документа также могут быть представлены фактами. Поэтому документ может быть материализован как концепт специального типа Document («Документ»), а его содержимое и извлечённая информация остаются в одном контейнере. Практическая задача извлечения содержимого и логической структуры документов в унифицированное представление рассматривается, например, в системе Dedoc [14].

Определение 8 (База знаний). База знаний Z – это составной TDM-контейнер

$$Z = (id_Z, F_Z, D_Z, \theta),$$

где id_Z – непустое конечное множество глобальных идентификаторов базы знаний, F_Z – структурно замкнутое множество собственных фактов базы знаний, D_Z – конечное множество вложенных документов, а θ – предметная область базы знаний.



(a) *(b)*

Рис. 1. Основные элементы TDM: (a) модель документа; (b) модель базы знаний.
Fig. 1. Core Elements of TDM: (a) Document Model; (b) Knowledge Base Model.

Соотношение основных элементов документа и базы знаний в модели TDM показано на рис. 1. В документе сегменты связывают извлечённые значения с исходным содержимым, а в базе знаний концепты и их атрибуты образуют структурированное представление предметной информации.

Предметными фактами контейнера Z называются факты, описывающие содержательные объекты, свойства и отношения рассматриваемой предметной области; это множество обозначается $F_{\text{dom}}(Z)$. Служебные факты контейнера в него не включаются. Активные предметные факты образуют множество

$$F_{\text{act}}(Z) = \{f \in F_{\text{dom}}(Z) \mid s(f) \neq \text{deleted}\}.$$

Именно это множество используется при построении графовой проекции, нормальной формы и результата запроса, если явно не указан режим аудита. Различие между $F_{\text{dom}}(Z)$ и $F_{\text{act}}(Z)$ позволяет сохранять логически удалённые предметные факты для аудита, не считая их частью текущего знания базы.

Факты вложенных документов не обязаны физически входить в F_Z. Поэтому один и тот же идентификатор может использоваться для связи документного концепта с каноническим концептом базы знаний без немедленного физического слияния их атрибутов. Ограничение

единственного владельца идентификатора применяется внутри каждого материализованного контейнера фактов.

Если множество вложенных документов несущественно для локального рассуждения, далее используется сокращённая запись (id_Z, F_Z, θ), в которой предполагается $D_Z = \emptyset$ или фиксированное неизменяемое множество вложенных документов.

Определение 9 (Глобальное пространство идентификаторов). Пусть $Z = (id_Z, F_Z, D_Z, \theta)$ – база знаний, а I – множество внутренних идентификаторов фактов. Пространством идентификаторов базы знаний называется

$$\mathrm{IDSpace}(Z) = id_Z \times I.$$

Каждый идентификатор, впервые созданный в Z, имеет вид $(\iota_Z, \iota_f) \in id_Z \times I$. При переносе факта в другую базу знаний исходные идентификаторы сохраняются в $id(f)$.

Определение 10 (Согласованное идентификационное объединение). Для фактов f_1, f_2 одного вида и одного типа обозначим через $f_1 \sqcup f_2$ факт, полученный объединением их непротиворечивых компонент. При этом

$$id(f_1 \sqcup f_2) = id(f_1) \cup id(f_2).$$

Тип результата равен общему типу объединяемых фактов. Статус выбирается согласованно с порядком жизненного цикла факта; если один из фактов имеет статус «удалён» (deleted), результат также считается логически удалённым.

3.4 Семантическая эквивалентность

Семантическая эквивалентность используется для выявления дубликатов, подтверждения фактов и выполнения операции объединения. Она отличается от идентификационной эквивалентности: пересечение множеств $id(f)$ означает, что два представления уже имеют общий идентификационный класс, тогда как семантическая эквивалентность устанавливает, что факты описывают один и тот же объект, связь или свойство предметной области.

Операция $\mathcal{O}^Z_{check}$ является точкой, в которой формальная модель обращается к внешней семантике. Она может быть реализована экспертом, правилом, внешним справочником, программным алгоритмом или их композицией, но в модели рассматривается как результат проверки утверждения о реальном объекте, свойстве или отношении. До такой проверки факт может быть структурно корректным, но не считается подтверждённым знанием.

Предположение 1 (Корректность экспертного оракула проверки). Для фиксированного контекста проверки операция $\mathcal{O}^Z_{check}$, используемая для концептов, является тотальной, детерминированной и согласованной с отношением «описывает один и тот же объект реального мира». Это соответствует режиму, где существует компетентный эксперт или доверенный источник истинности. В частности, на концептах одного типа операция индуцирует рефлексивное, симметричное и транзитивное отношение.

Определение 11 (Семантическая эквивалентность концептов). Концепты $c_1, c_2 \in F_c$ семантически эквивалентны, если они имеют один тип и описывают один и тот же объект или понятие:

$$c_1 \equiv c_2 \Leftrightarrow t(c_1) = t(c_2) \wedge (id(c_1) \cap id(c_2) \neq \emptyset \vee \mathcal{O}^Z_{check}(c_1, c_2) = \text{true}).$$

Определение 12 (Семантическая эквивалентность аннотаций). Аннотации $a_1, a_2 \in F_a$ семантически эквивалентны, если они являются атрибутами семантически эквивалентных владельцев, имеют один тип и равные нормализованные значения:

$$a_1 \equiv a_2 \Leftrightarrow m(a_1) \equiv m(a_2) \wedge t(a_1) = t(a_2) \wedge \mathcal{O}^{\varphi}_{\text{normalize}}(a_1) = \mathcal{O}^{\varphi}_{\text{normalize}}(a_2).$$

Здесь $\mathcal{O}^{\varphi}_{\text{normalize}}$ – операция нормализации значения факт a определённая в разделе 6; если такая операция не задана, используется исходное значение.

Определение 13 (Семантическая эквивалентность групп и связей). Группы считаются семантически эквивалентными, если они имеют семантически эквивалентных владельцев,

один тип и взаимно покрывающие друг друга множества непосредственных атрибутов. Связи считаются семантически эквивалентными, если имеют один тип, взаимно эквивалентные обычные атрибуты и эквивалентные множества ролевых участников.

Утверждение 1 (Согласованность семантической эквивалентности). При выполнении предположения 1 отношение $\equiv$ является отношением эквивалентности на каждом из множеств F_a, F_g, F_c, F_r, где оно определено.

Доказательство. Для концептов утверждение следует непосредственно из предположения о корректности оракула проверки и равенства типов. Для аннотаций рефлексивность, симметричность и транзитивность следуют из соответствующих свойств эквивалентности владельцев, равенства типов и равенства нормализованных значений. Для групп и связей рассуждение проводится по структурной индукции по конечным деревьям атрибутов: взаимное покрытие атрибутов сохраняет свойства отношения эквивалентности, а ролевые участники связей сравниваются через эквивалентность соответствующих концептов.

4. Корректность TDM-контейнера

Целостностная часть TDM задаёт класс допустимых состояний контейнера фактов. Её задача состоит в том, чтобы гарантировать корректную форму фактов, разрешимость ссылок, согласованность атрибутной структуры, корректность ролей связей и соответствие схеме выбранной предметной области. Такой контейнер можно безопасно передавать между компонентами, нормализовать, отображать в другую предметную область и использовать для построения графовой проекции.

Семантическое подтверждение информации задаётся отдельной частью модели: статусами фактов и операциями $\mathcal{O}^Z_{check}$ и $\mathcal{O}^Z_{approve}$. Предикат WF_θ отвечает на вопрос, является ли контейнер допустимым состоянием модели, а соответствие факта реальному объекту или событию проверяется в фиксированном контексте внешней верификации.

В статье используется компактная форма предиката корректности. Подробные ограничения группируются по смысловым классам: идентификаторы, вид и тип, локальные предикаты типов, атрибутная структура, сегменты, связи, контейнерная замкнутость и корректность схемы.

Если читать это без формального аппарата, дальнейший предикат отвечает на практический вопрос: можно ли передать контейнер следующему компоненту без потери ссылок, нарушения типов, разрыва атрибутов или некорректных ролей связей. Истинность сведений о внешнем мире при этом проверяется отдельно.

Определение 14 (Структурная корректность). Пусть $Z = (id_Z, F, D, \theta)$ – документ или база знаний в предметной области θ. TDM-контейнер называется структурно корректным относительно θ, если выполняется предикат

$$\mathrm{WF}_\theta(Z) \Leftrightarrow I_{\text{id}} \wedge I_{\text{kind}} \wedge I_{\text{type}} \wedge I_{\text{attr}} \wedge I_{\text{seg}} \wedge I_{\text{rel}} \wedge I_{\text{container}} \wedge I_{\text{schema}}.$$

Здесь I_{id} задаёт единственность владельца идентификатора и нормализацию идентификационных классов; I_{kind} – совпадение вида факта и множества типов; I_{type} – выполнение локальных предикатов целостности типов; I_{attr} – ацикличность атрибутной структуры, соответствие атрибутных типов сигнатуре и кратности; I_{seg} – корректность сегментов и их ссылок; I_{rel} – корректность ролевых участников связей; $I_{\text{container}}$ – замкнутость контейнера относительно структурных ссылок; I_{schema} – корректность схемы предметной области.

Определение 15 (Целостность базы знаний). База знаний находится в целостном состоянии тогда и только тогда, когда она структурно корректна относительно своей предметной области:

$$\text{consistent}_\theta(Z) \Leftrightarrow \mathrm{WF}_\theta(Z).$$

Определение 16 (Корректность факта в контексте). Пусть f – факт, а $Z = (id_Z, F, D, \theta)$ – база знаний. Факт называется корректным в контексте Z, если добавление f к контейнеру с учётом возможного идентификационного объединения сохраняет структурную корректность:

$$\text{valid}_\theta(f; Z) \Leftrightarrow \text{WF}_\theta(id_Z, F \cup \{f\}, D, \theta).$$

Если идентификаторы f пересекаются с идентификаторами фактов из F, проверяется корректность результата их согласованного объединения.

Перечисленные компоненты являются не самостоятельной теоремой, а декомпозицией определения WF_θ. Она отделяет структурную часть модели от манипуляционной части: любая операция, изменяющая контейнер, должна либо вернуть новое состояние, удовлетворяющее этой конъюнкции ограничений, либо завершиться отказом. Прикладные требования, такие как временная согласованность, правила доступа или специализированные бизнес-правила, задаются дополнительно в конкретной предметной области.

5. Граф знаний, покрытие и деревья раскрытия

Граф знаний отражает семантические отношения между аннотациями, группами, концептами и связями. Сегменты в него не входят: они фиксируют происхождение значения аннотации и рассматриваются ограничениями корректности отдельно. Такое разделение позволяет использовать граф знаний как семантический индекс, а полный контейнер фактов – как источник доказательств и восстановления происхождения значений.

Определение 17 (Граф знаний). Графом знаний базы знаний Z называется неориентированный типизированный граф

$$G_Z = (F^{\text{sem}}, E), \qquad F^{\text{sem}} = \{f \in F_a \cup F_g \cup F_c \cup F_r \mid s(f) \neq \text{deleted}\},$$

где ребро $\{f_i, f_j\} \in E$ существует тогда и только тогда, когда выполнено одно из условий: $f_j \in A(f_i)$, где $f_i \in F_g \cup F_c \cup F_r$ и $f_j \in F_a \cup F_g$; $f_i \in F_r$, а $f_j \in \text{Participants}(f_i)$.

Роли участников связи не являются свойствами ребра графа. Они хранятся как ролевые атрибуты факта-связи и задаются сигнатурой его типа. Поэтому неориентированный граф сохраняет семантическую смежность фактов, не дублируя ролевую структуру факта-связи.

Граф предметной области θ задаёт допустимые смежности типов:

$$G^\theta = (T_\theta^{\text{sem}}, E_\theta), \qquad T_\theta^{\text{sem}} = T_a \cup T_g \cup T_c \cup T_r.$$

Ребро $\{t, u\} \in E_\theta$ существует, если тип u допустим, как обычный атрибут типа t, либо если t является типом связи, а u допустим, как тип концепта-участника одной из ролей этой связи.

Теорема 1 (Согласованность графа и предметной области). Если $\text{WF}_\theta(Z)$, то граф знаний G_Z структурно соответствует предметной области θ : каждое его атрибутное ребро разрешено атрибутной сигнатурой владельца, а каждое ребро между связью и концептом-участником – ролевой сигнатурой типа связи.

Доказательство непосредственно следует из компонент I_{attr}, I_{rel} и I_{kind} предиката структурной корректности.

Локальная структура концепта или связи в TDM является деревом атрибутов. Это свойство важно для обработки документов: даже если глобальный граф знаний содержит циклы через связи между концептами, атрибутное описание каждого отдельного концепта или связи остаётся конечной древовидной структурой.

Определение 18 (Дерево концепта и расширенное дерево связи). Для концепта $c \in F_c$ определим дерево его атрибутов:

$$U(c) = G_Z[\{c\} \cup A^+(c)].$$

Для связи $r \in F_r$ определим расширенное дерево:

$$U(r) = G_Z[\{r\} \cup \text{Participants}(r) \cup A^+(r)].$$

Далее для $f \in F_c \cup F_r$ запись $U(f)$ означает соответствующее дерево концепта или расширенное дерево связи.

Примеры локальных подграфов концепта и связи показаны на рис. 2: атрибуты образуют древовидную структуру относительно корневого факта, а участники связи присоединяются к корню через ролевые атрибуты.

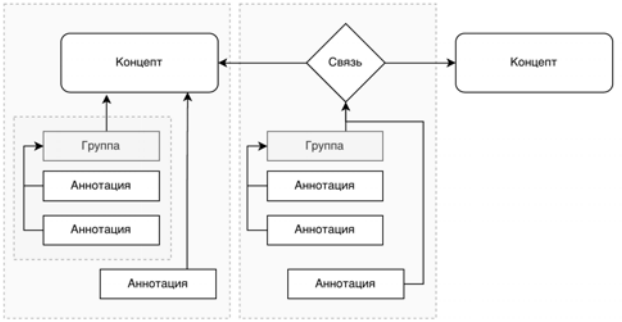


Рис. 2. Локальные подграфы концептов и связей в TDM.
Fig. 2. Local Subgraphs of Concepts and Relations in TDM.

Теорема 2 (Древовидность локальных подграфов). Если $\text{WF}_\theta(Z)$, то $U(c)$ для любого $c \in F_c$ и $U(r)$ для любого $r \in F_r$ являются конечными корневыми деревьями с корнями c и r соответственно.

Доказательство опирается на два свойства корректного контейнера: каждый атрибут имеет единственного владельца, а отношение владения атрибутом ациклично. Поэтому цепочка владельцев любой аннотации или группы конечна и заканчивается в некотором концепте или связи; участники связи добавляются к расширенному дереву только через корень связи и не создают атрибутного цикла.

Теорема 3 (Покрытие графа знаний деревьями атрибутов). Если $\text{WF}_\theta(Z)$, то семейство

$$C_Z = \{\, U(c) \mid c \in F_c \} \cup \{\, U(r) \mid r \in F_r \}$$

образует покрытие графа знаний G_Z.

Каждая вершина $F_c \cup F_r$ входит в собственное дерево, а каждая аннотация или группа попадает в дерево ближайшего владельца из $F_c \cup F_r$. Атрибутные рёбра покрываются деревом соответствующего корня, а рёбра участия – расширенным деревом связи.

Граф предметной области может содержать циклы и допускает повторное использование одного и того же типа группы в разных структурных контекстах. Поэтому предметную область нельзя всегда отождествлять с единственным деревом типов. Вместо этого TDM использует конечные раскрытия типов, индуцированные конкретной базой знаний.

Определение 19 (Дерево раскрытия типа). Пусть $t \in T_c \cup T_r$. Рассмотрим все пути от корня в деревьях $U(f)$, где $f \in F_c \cup F_r$ и $t(f) = t$. Если путь имеет вид

$$f_0, f_1, \ldots, f_k, \qquad f_0 = f,$$

то каждому его начальному отрезку сопоставляется последовательность типов

$$p_k = (t(f_0), t(f_1), \ldots, t(f_k)).$$

Деревом раскрытия типа t относительно базы знаний Z называется корневое дерево

$$T_Z(t) = (V_t^{\text{unf}}, E_t^{\text{unf}}),$$

в котором вершинами являются вхождения $(p_k, t(f_k))$, а рёбра соединяют вхождения, соответствующие соседним начальным отрезкам одного пути.

Одна и та же метка типа может встречаться в нескольких вершинах $T_Z(t)$, если она достигается разными путями. Поэтому вершины дерева раскрытия являются не только типами, а вхождениями типов. Проекция $\pi_t(p,u) = u$ сопоставляет каждому такому вхождению исходный тип предметной области.

Теорема 4 (Существование и конечность дерева раскрытия). Если $\mathrm{WF}_\theta(Z)$, то для каждого типа $t \in T_c \cup T_r$, реализованного в Z, дерево $T_Z(t)$ существует и конечно.

Дерево раскрытия строится по конечному семейству конечных деревьев $U(f)$, покрывающих граф знаний. В каждом таком дереве число путей от корня конечно, следовательно конечно и множество всех начальных отрезков путей, образующих $T_Z(t)$.

Теорема 5 (Согласованность раскрытия с графом предметной области). Если $\mathrm{WF}_\theta(Z)$, то проекция π_t является гомоморфизмом из дерева раскрытия $T_Z(t)$ в граф предметной области G^θ: для каждого ребра $\{x,y\} \in E_t^{\mathrm{unf}}$ выполняется

$$\{\pi_t(x), \pi_t(y)\} \in E_\theta.$$

При этом π_t не обязана быть инъективной.

Обоснование следует из теоремы 1: соседние вершины дерева раскрытия происходят либо из отношения атрибута, либо из участия концепта в связи, а оба вида смежности разрешены графом предметной области.

Покрытия деревьями и связанные с ними формальные результаты далее используются при построении алгоритмов отображения баз знаний между предметными областями.

6. Операции TDM

Операции TDM изменяют контейнер фактов. Каждая операция рассматривается как частичная: если её результат нарушает WF_θ или предпосылки нормализации 2, операция не создаёт новое состояние, а возвращает отказ.

На сквозном примере эти операции соответствуют обычному жизненному циклу извлечения: создать контейнер, добавить аннотации Name, CityName и Date вместе с сегментами происхождения, проверить совпадение извлечённого человека с известным объектом, подтвердить или отклонить факты и затем нормализовать результат.

- $\mathcal{O}_{\mathrm{create}}(\theta)$. Назначение: создать пустую базу знаний в предметной области θ. Вход: предметная область. Выход: $Z = (id_Z, \emptyset, D, \theta)$ с новым непустым множеством идентификаторов id_Z. Допустимость: схема θ удовлетворяет I_{schema}. Эффект: создаётся корректное начальное состояние контейнера.
- $\mathcal{O}^Z_{add}(f)$. Назначение: добавить факт-кандидат в базу знаний. Вход: $Z = (id_Z, F, D, \theta)$ и факт f. Выход: новая база Z'. Допустимость: f имеет допустимый вид и тип, а результат добавления с учётом возможного идентификационного объединения удовлетворяет WF_θ. Эффект: если идентификаторы f не пересекаются с F, факт добавляется; иначе вся идентификационная компонента заменяется согласованным объединением.
- $\mathcal{O}^Z_{addAll}(X)$. Назначение: добавить конечное множество фактов. Вход: база знаний Z и конечное множество X. Выход: результат последовательного применения $\mathcal{O}^Z_{add}$ ко всем фактам из X. Допустимость: каждое промежуточное состояние корректно или приводится к корректному состоянию идентификационным объединением. Эффект: используется при построении проекций и расширений; если порядок добавления влияет на промежуточные состояния, далее сравниваются приведённые нормальные формы результатов.
- $\mathcal{O}^Z_{update}(f, f')$. Назначение: заменить факт новой версией без изменения его идентификационного класса. Вход: $f \in F$ и f' с $id(f') = id(f)$. Выход: база Z'.

Допустимость: замена сохраняет WF_θ. Эффект: меняются статус, значение или структурные составляющие факта, но не его идентификаторы.

- $\mathcal{O}^Z_{delete}(f)$ и $\mathcal{O}^Z_{logical-delete}(f)$. Назначение: удалить факт физически или перевести его в статус deleted. Вход: факт $f \in F$. Выход: база Z'. Допустимость: вместе с f обрабатываются все зависимые факты, необходимые для сохранения WF_θ. Эффект: физическое удаление исключает факты из контейнера, а логическое удаление сохраняет их как часть истории обработки.
- $\mathcal{O}^Z_{check}(f, x)$. Назначение: проверить истинность или семантическое соответствие факта в фиксированном контексте. Вход: проверяемый факт f и объект сопоставления x, например концепт, связь, документальное свидетельство или внешний источник. Выход: true или false. Допустимость: контекст проверки фиксирован. Эффект: операция не меняет контейнер, но связывает формальный факт с утверждением о реальном мире.
- $\mathcal{O}^Z_{approve}(f)$. Назначение: материализовать результат проверки в контейнере. Вход: факт $f \in F$. Выход: база Z'. Допустимость: $\mathcal{O}^Z_{check}$ определена для требуемых сопоставлений. Эффект: факт переводится в статус approved либо удаляется; каскадные изменения выполняются через $\mathcal{O}^Z_{update}$, $\mathcal{O}^Z_{delete}$ или $\mathcal{O}^Z_{logical-delete}$.
- $\mathcal{O}^Z_{merge}(c_1, c_2)$. Назначение: объединить семантически эквивалентные концепты. Вход: $c_1, c_2 \in F_c$. Выход: база Z'. Допустимость: концепты имеют один тип и удовлетворяют $\mathcal{O}^Z_{check}(c_1, c_2) = \mathrm{true}$ либо имеют пересекающиеся идентификаторы. Эффект: концепты заменяются одним представителем, их идентификаторы объединяются, а эквивалентные атрибуты и связанные роли объединяются рекурсивно.
- $\mathcal{O}^Z_{split\text{-}concept}(c, \alpha, \beta)$. Назначение: разделить концепт, в котором смешаны сведения о разных объектах. Вход: концепт c, разбиение атрибутов α и разбиение инцидентных связей β. Выход: база Z'. Допустимость: распределение атрибутов и связей сохраняет WF_θ. Эффект: создаётся новый концепт того же типа, часть атрибутов и связей переносится к нему.
- $\mathcal{O}^Z_{rebind}(c, c_{\mathrm{old}}, c_{\mathrm{new}})$. Назначение: перепривязать концепт к другому идентификационному классу. Вход: перепривязываемый концепт и старая/новая опорные идентичности. Выход: база Z'. Допустимость: замена идентификатора не создаёт противоречивого объединения. Эффект: используется после разделения или при синхронизации контейнеров.
- $\mathcal{O}^\varphi_{\mathrm{aggregate}}(X)$. Назначение: вычислить новое значение по конечному набору аннотаций. Вход: конечное $X \subseteq F_a$ и функция агрегации φ. Выход: аннотация-кандидат или отказ. Допустимость: φ определена для типов входных значений и задаёт допустимые тип и владельца результата. Эффект: создаётся новый факт-кандидат; добавление в контейнер выполняет $\mathcal{O}^Z_{add}$.
- $\mathcal{O}^\varphi_{\mathrm{normalize}}(f_a)$. Назначение: привести значение аннотации к каноническому виду типа. Вход: аннотация f_a и функция нормализации φ, заданная предметной областью. Выход: аннотация с теми же идентификаторами, типом, владельцем и статусом, но с нормализованным значением, либо отказ. Допустимость: нормализованное значение принадлежит области допустимых значений типа. Эффект: операция используется при проверке семантической эквивалентности аннотаций и при выводе фактов целевой области.
- $\mathcal{O}^Z_{select}(\varphi)$ и $\mathcal{O}^Z_{subset}(\varphi)$. Назначение: получить факты или подграф, удовлетворяющие поисковому запросу φ. Вход: запрос как конечное правило обхода графовой

проекции. Выход: множество фактов или индуцированный подграф. Эффект: операции не изменяют базу знаний и используются для отбора исходных фактов при построении презентаций и междоменных отображений.

Определение 20 (Операции получения значений). Операциями получения значений называются конечные операции $\mathcal{O}^Z_{select}$, $\mathcal{O}^Z_{subset}$, $\mathcal{O}_{\text{aggregate}}$ и $\mathcal{O}_{\text{normalize}}$, которые выбирают исходные факты, вычисляют значения целевых фактов и не создают произвольных новых утверждений.

Определение 21 (Выводимый факт). Пусть $Z = (id_Z, F, \theta)$ – база знаний предметной области θ, а f – факт некоторой, возможно другой, предметной области. Факт f называется выводимым из Z, что обозначается $\text{derive}(Z, f)$, если существует конечная презентация, задающая тип, структурную позицию и значения факта. Для аннотации вычисляется её значение через операции получения значений из фактов F; для группы, концепта или связи – значения всех аннотативных листьев дерева атрибутов, а роли и владельцы задаются структурным шаблоном презентации.

Определение 22 (Презентация факта). Презентацией факта f относительно базы знаний Z называется конечное описание

$$p = (\text{source}, \text{shape}, \text{compute}),$$

где source задаёт исходные факты, shape – целевой тип, владельца, роли и дерево атрибутов, а compute – вычисление значений аннотаций. Идентификаторы создаются операцией добавления или сохраняются из источников как происхождение идентификационного класса и не считаются новой предметной информацией. Если для f существует такая презентация, то $\text{derive}(Z, f)$.

Операции произвольного добавления, обновления и изменения статуса не входят в выводимость: они меняют контейнер, но не задают вычислимость предметного содержания.

6.1 Сквозной пример

Рассмотрим фразу, введенную выше. В предметной области θ_{bio} фрагмент может быть материализован следующими фактами.

Графовая проекция содержит c_1, c_2, r_1 и их атрибуты, но не содержит сегменты s_1, s_2 и s_3: они нужны для восстановления происхождения значений a_1, a_2 и a_5. Если проверка $\mathcal{O}^Z_{check}$ подтверждает, что c_1 совпадает с уже известным концептом «Петров Иван», операция $\mathcal{O}^Z_{merge}$ объединяет идентификаторы и атрибуты.

7. Приведённая нормальная форма

Предположение 2 (Предпосылки нормализации). Нормализация рассматривается в классе детерминированных TDM-конвейеров при следующих предпосылках: множество фактов контейнера конечно, а все деревья атрибутов и деревья покрытия, построенные по этому множеству, конечны; операция $\mathcal{O}^Z_{check}$ в фиксированном контексте проверки тотальна, детерминирована и согласована с семантической эквивалентностью; для любого факта, который не может быть подтверждён, множество фактов, удаляемых вместе с ним из-за структурных зависимостей, является наименьшим множеством, замкнутым относительно зависимостей графа покрытия; согласованное объединение $\sqcup$, включая специальное объединение концептов операцией $\mathcal{O}^Z_{merge}$, сохраняет ограничения целостности и объединяет множества идентификаторов и эквивалентные атрибуты ассоциативно, коммутативно и идемпотентно с точностью до выбора представителя; для фиксированной конечной базы знаний и фиксированной целевой предметной области множество выводимых валидных целевых фактов конечно с точностью до нормализации и идентификационного объединения.

Иными словами, предположение 2 задаёт управляемый режим: данные конечны, проверка воспроизводима, отклонение удаляет только необходимые зависимости, объединение дублей не нарушает целостность, а нормализация и междоменный вывод завершаются на конечном множестве фактов. Ошибки экспертов, изменения внешнего мира, вероятностные оценки и галлюцинации LLM-акторов могут учитываться расширениями модели: журналом проверок, версиями контекста, оценками уверенности и политиками выбора альтернатив.

Определение 23 (Объединяемые концепты). Концепты $c_1, c_2 \in F_c$, $c_1 \neq c_2$, называются объединяемыми в базе знаний Z, если они имеют один тип и описывают один и тот же объект:

$$\text{mergeable}_Z(c_1, c_2) \Leftrightarrow t(c_1) = t(c_2) \wedge (\mathcal{O}^Z_{check}(c_1, c_2) = \text{true} \vee id(c_1) \cap id(c_2) \neq \emptyset).$$

Определение 24 (Нормальная форма базы знаний). База знаний $Z = (id_Z, F, D, \theta)$ находится в нормальной форме, если выполнены условия: $\text{WF}_\theta(Z) = \text{true}$; все активные предметные факты базы знаний подтверждены: $s(f) = \text{approved}$ для всех $f \in F_{\text{act}}(Z)$; для любых активных концептов $c_1, c_2 \in F_c$, если $c_1 \neq c_2$, то $\text{mergeable}_Z(c_1, c_2) = \text{false}$.

Последнее условие означает, что вся идентификационная и семантическая эквивалентность концептов уже разрешена операцией $\mathcal{O}^Z_{merge}$. Поэтому нормальная форма не содержит двух различных представителей одного и того же объекта реального мира. При этом нормальная форма не требует полноты предметной области: она фиксирует внутреннюю согласованность базы знаний, подтверждённость активных фактов и отсутствие неразрешённых дубликатов.

Определение 25 (Приведённая нормальная форма). База знаний $Z = (id_Z, F, D, \theta)$ называется приведённой нормальной формой базы знаний Z', если: Z находится в нормальной форме; каждый факт Z выводим из Z' ; F является максимальным по включению множеством фактов, удовлетворяющим условиям нормальности и достижимости. Приведённая нормальная форма обозначается $\text{RNF}(Z')$.

Будем называть нормализующими шагами два вида преобразований базы знаний: подтверждение фактов операцией $\mathcal{O}^Z_{approve}$ или отклонение операцией $\mathcal{O}^Z_{reject}$ с каскадным удалением неподтверждённых зависимых фактов, а также разрешение идентификационных классов фактов. Второй шаг включает обычное согласованное объединение фактов с пересекающимися множествами идентификаторов и специальное объединение концептов операцией $\mathcal{O}^Z_{merge}$, при котором рекурсивно объединяются эквивалентные атрибуты и перепривязываются связи. Операции разделения $\mathcal{O}^Z_{split\text{-}concept}$ и перепривязки $\mathcal{O}^Z_{rebind}$ не являются автоматическими шагами построения RNF: они фиксируют выбор актора или внешнего алгоритма и могут применяться до последующей нормализации.

Теорема 6 (Существование и единственность приведённой нормальной формы).
В классе детерминированных TDM-конвейеров, удовлетворяющих предпосылкам нормализации 2, для любой конечной базы знаний Z существует единственная приведённая нормальная форма. Она может быть получена любой конечной последовательностью нормализующих шагов, доведённой до неподвижной точки.

8. Отображение предметных областей и медиаторы

Этот раздел является центральным для предлагаемой модели. Предыдущие разделы описывали состояние одного контейнера; теперь вводится формальный способ перенести содержание между разными предметными областями так, чтобы результат снова был корректной базой знаний, а не набором неформальных сообщений.

8.1 Компоненты обработки

Определение 26 (Компонент обработки). Компонент обработки – полное отображение между TDM-контейнерами:

$$P_{\theta_1 \to \theta_2} : Z_{\theta_1} \to Z_{\theta_2}.$$

Компоненты обработки могут быть реализованы программными алгоритмами, человеком-оператором, внешними сервисами, языковыми моделями или их композициями. Для модели важно только то, что входом и выходом является контейнер фактов с определённой предметной областью.

Определение 27 (Актор). Актор – компонент обработки, действующий в пределах одной предметной области θ :

$$\mathbb{A}_\theta : (id_Z, F, \theta) \to (id_Z, F', \theta).$$

Множество F' получается из F применением допустимых операций манипуляционной части. Если актор неприменим, то $F' = F$.

Определение 28 (Медиатор). Медиатор – компонент обработки, осуществляющий отображение документа или базы знаний из предметной области θ_1 в область θ_2:

$$\mu_{\theta_1 \to \theta_2} : Z_{\theta_1} \to Z_{\theta_2}.$$

Результатом работы медиатора является TDM-контейнер, интерпретированный в терминах целевой предметной области.

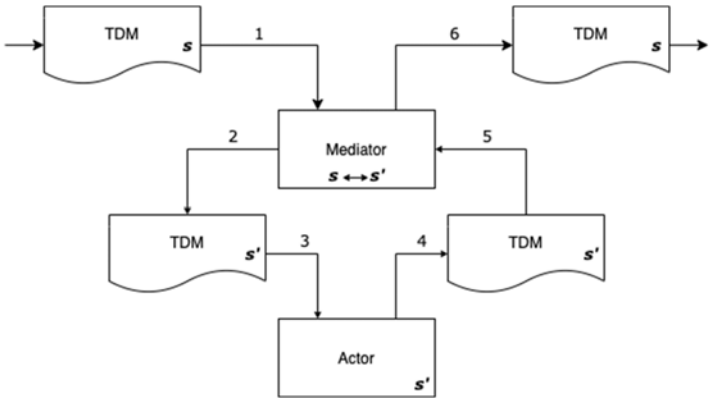


Рис. 3. Использование медиатора в процессе обработки.
Fig. 3. Use of a Mediator in the Processing Workflow.

Общая схема использования медиатора при переносе информации между предметными областями показана на рис. 3. Медиатор решает семантическую задачу переноса знаний. Транспортный протокол может определить, как передать сообщение или вызвать инструмент, но не задаёт, какие факты считаются эквивалентными, как переносить атрибуты и связи между предметными областями и как сохранять корректность результата.

В примере выше медиатор в область θ_{profile} может заменить связь BornIn атрибутами BirthPlace и BirthDate концепта Profile; обратное отображение не полно, если в целевой области не хранится отдельный концепт города.

8.2 Проекции на предметную область

В этом разделе используется локальная форма корректности факта:

$$\text{valid}_\theta(f) \Leftrightarrow \exists\, (id_Z, F_Z, D_Z, \theta) : f \in F_Z \wedge \text{WF}_\theta(id_Z, F_Z, D_Z, \theta).$$

Она означает, что факт может входить хотя бы в один конечный корректный контейнер предметной области θ. В отличие от $\text{valid}_\theta(f; Z)$, введённой в определении 16, локальная корректность не требует совместимости факта с заранее заданной базой знаний.

Интуитивно проекция отвечает не на вопрос «как физически скопировать записи», а на вопрос «какое предметное содержание исходной базы можно корректно выразить в целевой области». Поэтому дальше различаются выводимость факта, его валидность в целевой схеме и представленность в уже построенной проекции.

Определение 29 (Выводимые валидные факты целевой области). Пусть $Z_1 = (id_{Z_1}, F_1, D_1, \theta_1)$ – база знаний, а θ_2 – целевая предметная область. Множеством выводимых валидных фактов целевой области называется

$$D_{\theta_2}(Z_1) = \{f \mid \text{derive}(Z_1, f) \wedge \text{valid}_{\theta_2}(f)\}.$$

Будем предполагать совместимость выводимого целевого замыкания: добавление любого $f \in D_{\theta_2}(Z_1)$ к любой проекции Z_1 на θ_2 с последующей нормализацией не отклоняется как конфликт целостности. Если факт уже представлен в нормализованной проекции, предметное содержание не меняется; если не представлен, оно строго расширяется.

Определение 30 (Проекция базы знаний). База знаний $Z_2 = (id_{Z_2}, F_2, D_2, \theta_2)$ называется проекцией базы знаний Z_1 на предметную область θ_2, если

$$\forall\, f \in F_{\text{act}}(Z_2): \quad f \in D_{\theta_2}(Z_1).$$

Порядок на проекциях задаётся включением их нормализованного предметного содержания:

$$Z_a \preccurlyeq Z_b \Leftrightarrow F_{\text{act}}(\text{RNF}(Z_a)) \subseteq F_{\text{act}}(\text{RNF}(Z_b)).$$

Проекция называется полной, если она является максимальным элементом этого порядка.

Теорема 7 (Свойства полных проекций). Пусть Z_1 – конечная база знаний, θ_2 – предметная область, и выполнены предпосылки конечности вычислимого замыкания и совместимости выводимого целевого замыкания. Тогда: существует по крайней мере одна полная проекция Z_1 на θ_2; приведённая нормальная форма полной проекции единственна с точностью до выбора идентификатора контейнера; добавление выводимого факта к проекции с последующей нормализацией снова даёт проекцию.

8.3 Операционные правила и отображения

Определение 31 (Операционное правило отображения). Операционным правилом отображения из предметной области θ_1 в предметную область θ_2 называется конечная композиция операций манипуляционной части, которая для базы знаний $Z_1 = (id_{Z_1}, F_1, D_1, \theta_1)$ возвращает конечное множество фактов целевой предметной области:

$$r(Z_1) \subseteq D_{\theta_2}(Z_1).$$

Реализационная форма правила может включать целевой тип, позицию в целевой структуре, запрос источников и вычислительное выражение для значений.

Определение 32 (Отображение предметных областей). Пусть θ_1 и θ_2 – предметные области. Отображением предметной области θ_1 в предметную область θ_2 называется конечное множество операционных правил

$$T_{\theta_1 \to \theta_2} = \{r_1, \ldots, r_n\}.$$

Результатом применения отображения к базе знаний Z_1 называется

$$T_{\theta_1 \to \theta_2}(Z_1) = \bigcup_{r \in T_{\theta_1 \to \theta_2}} r(Z_1).$$

Определение 33 (Представленность факта в проекции). Пусть Z_2 – проекция Z_1 на θ_2, а $f \in D_{\theta_2}(Z_1)$. Факт f представлен в Z_2, если его добавление к Z_2 не расширяет нормализованное предметное содержание:

$$F_{\text{act}}(\text{RNF}(Z_2)) = F_{\text{act}}(\text{RNF}(\mathcal{O}^{Z_2}_{addAll}(\{f\}))).$$

Определение 34 (Полнота отображения относительно базы знаний). Отображение $T_{\theta_1 \to \theta_2}$ называется полным относительно базы знаний Z_1 и предметной области θ_2, если для базы знаний

$$Z_T = \text{RNF}\ (\ \mathcal{O}_{addAll}^{(id_T, \emptyset, \theta_2)}\quad (\ T_{\theta_1 \to \theta_2}(Z_1)))$$

каждый факт $f \in D_{\theta_2}(Z_1)$ представлен в Z_T.

Теорема 8 (Полное отображение и полная проекция). При предпосылках теоремы 7 существует конечное отображение $T_{\theta_1 \to \theta_2}$, полное относительно Z_1 и θ_2. Если такое отображение задано и

$$Z_2 = \text{RNF}\ (\ \mathcal{O}_{addAll}^{(id_{Z_2}, \emptyset, \theta_2)}\quad (\ T_{\theta_1 \to \theta_2}(Z_1))).$$

то Z_2 является полной проекцией базы знаний Z_1 на предметную область θ_2.

Существование имеет неконструктивный смысл: для каждого факта полной проекции можно выбрать конечную композицию операций, порождающую этот факт. Достаточность следует из определения полноты отображения: любой выводимый валидный факт уже представлен в $\text{RNF}(Z_2)$, поэтому другая проекция не может строго расширить нормализованное предметное содержание Z_2.

8.4 Алгоритм медиатора

Медиатор реализует отображение предметных областей. Для фиксированного отображения $T_{\theta_1 \to \theta_2}$ соответствующий медиатор обозначается

$$\mu_{\theta_1 \to \theta_2}^{T}: Z_{\theta_1} \to Z_{\theta_2}.$$

Последовательность применения правил отображения и построения целевой базы знаний приведена в алгоритме 1.

	Вход: База знаний $Z_1 = (id_{Z_1}, F_1, D_1, \theta_1)$, отображение $T_{\theta_1 \to \theta_2}$, целевая предметная область θ_2
	Выход: База знаний $Z_2 = (id_{Z_2}, F_2, D_2, \theta_2)$
1	$F_2 \leftarrow \emptyset$
2	`для кiвсех` $r \in T_{\theta_1 \to \theta_2}$ `выполнить`
3	$G \leftarrow r(Z_1)$
4	`для всех` $f \in G$ `выполнить`
5	$(id_{Z_2}, F_2, \theta_2) \leftarrow \mathcal{O}_{add}^{(id_{Z_2}, F_2, \theta_2)}(f)$
6	$Z_2 \leftarrow \text{RNF}((id_{Z_2}, F_2, \emptyset, \theta_2))$

Алгоритм 1. Медиатор, реализующий отображение предметных областей.
Algorithm 1. Mediator Implementing Domain Mapping.

Если отображение не порождает ни одного предметного факта, алгоритм возвращает тривиальную проекцию целевой предметной области. Поэтому медиатор остаётся тотальным отображением между контейнерами. Если $T_{\theta_1 \to \theta_2}$ полно относительно Z_1 и θ_2, то алгоритм 1 возвращает полную проекцию по теореме 8.

8.5 Расширение базы знаний информацией из другой области

При расширении одной базы знаний фактами, полученными из другой базы, используется глобальное пространство идентификаторов, заданное в определении 9. Если $Z_1 = (id_{Z_1}, F_1, D_1, \theta_1)$ и $Z_2 = (id_{Z_2}, F_2, D_2, \theta_2)$ – разные базы знаний, причём

$$id_{Z_1} \cap id_{Z_2} = \emptyset.$$

то идентификаторы фактов этих баз знаний не пересекаются:

$$\forall f_1 \in F_1, \forall f_2 \in F_2: id(f_1) \cap id(f_2) = \emptyset.$$

Действительно, по определению пространства идентификаторов

$$id(f_1) \subseteq id_{Z_1} \times I, \qquad id(f_2) \subseteq id_{Z_2} \times I.$$

Если бы существовал идентификатор $(\iota, \eta) \in id(f_1) \cap id(f_2)$, то его первая компонента одновременно принадлежала бы id_{Z_1} и id_{Z_2}, что противоречит условию $id_{Z_1} \cap id_{Z_2} = \emptyset$.

Пусть имеются базы знаний Z_1 и Z_2. Построим проекцию $Z_1^{\theta_2} = \mu_{\theta_1 \to \theta_2}^{T}(Z_1)$ в предметную область θ_2. Расширением Z_2 информацией из Z_1 назовём базу знаний

$$Z_2^{+} = \text{RNF}\ (\ \mathcal{O}_{addAll}^{Z_2}(F(Z_1^{\theta_2}))).$$

Если $Z_1^{\theta_2}$ является полной проекцией Z_1 на θ_2, то расширение Z_2^{+} содержит информацию из Z_2 и всю информацию из Z_1, выводимую в предметной области θ_2, с точностью до приведённой нормальной формы. Единственность такого расширения следует из теоремы 6: любые порядки добавления, подтверждения и объединения имеют одно и то же исходное множество фактов Z_2 и полной проекции $Z_1^{\theta_2}$.

Тем самым раздел задаёт формальный смысл междоменного переноса в TDM. Проекция описывает, какие факты целевой области выводимы из исходной базы; полнота фиксирует отсутствие потерь выводимой информации; отображение задаёт конечный способ построения проекции; медиатор реализует это отображение как преобразование контейнеров. Поэтому результат медиатора можно рассматривать не как эвристический обмен сообщениями, а как корректную и, при полном отображении, полную базу знаний целевой предметной области.

9. Обсуждение и ограничения

Предложенная модель задаёт формальный слой состояния, но не заменяет алгоритмы извлечения, разрешения многозначности, проверки и синтеза отображений. Корректность TDM-контейнера означает структурную корректность и сохранение инвариантов, а не автоматическую истинность всех извлечённых утверждений. Теоремы о нормализации и полных проекциях сформулированы для детерминированных конвейеров с фиксированным контекстом проверки и согласованной политикой разрешения конфликтов.

Если проверка вероятностна, эксперты расходятся или внешний мир меняется во времени, модель дополняется журналом проверок, версиями контекста, оценками уверенности и политиками выбора. Существование полного отображения также не является алгоритмом синтеза медиатора: на практике полнота проверяется относительно конкретного набора типов, запросов и операций. RDF/OWL, SHACL, PROV, графовые базы данных и реляционные схемы при этом могут использоваться как форматы реализации отдельных проекций TDM.

10. Заключение

В работе предложена формальная модель TDM, рассматривающая документ и базу знаний как контейнеры типизированных фактов. Единая форма факта позволяет представлять исходный контент, извлечённые утверждения, происхождение значений, статусы проверки,

графовую проекцию и перенос информации между предметными областями в одном формальном слое.

Для модели введены структурные ограничения, приведённая нормальная форма, графовая проекция и медиаторы переноса информации между предметными областями. Полученные свойства показывают, что результат работы медиатора может рассматриваться не как произвольное преобразование сообщений, а как корректная и, при полном отображении, полная база знаний целевой предметной области. Дальнейшая работа связана с языком правил отображения, эталонной сериализацией TDM-документов и экспериментальной оценкой модели в RAG-, GraphRAG- и агентных конвейерах.

Программная реализация модели TDM представлена в открытом репозитории [15].

Список литературы / References

[1]. Турдаков Д. Ю., Аветисян А.И. и др. Доверенный искусственный интеллект: вызовы и перспективные решения. Доклады Российской академии наук. Математика, информатика, процессы управления, 2022, т. 508(0), с. 13-18.

[2]. Колокольников Ф.А., Орлов В.В., Турдаков Д.Ю. Перспективы использования доверенной информационной аналитической системы на базе платформы Талисман с применением методов искусственного интеллекта для повышения эффективности эксплуатации сложных аппаратных систем. Труды Института системного программирования РАН, 2024, т. 36, вып. 3, с. 106-22.

[3]. Euzenat J., Shvaiko P. Ontology Matching. Springer Berlin Heidelberg, Berlin, Heidelberg, 2d ed., 2013.

[4]. Fagin R., Kolaitis P.G., Miller R.J., Popa L. Data exchange: Semantics and query answering. Theoretical Computer Science, 2005, 336(1), pp. 89-124.

[5]. Hartig O. Foundations of RDF* and SPARQL*. arXiv preprint, 2017. DOI: 10.48550/arXiv.1703.07936.

[6]. Knublauch H., Kontokostas D. Shapes constraint language (SHACL). W3C Recommendation, 2017.

[7]. Moreau L., Missier P. PROV-DM: The PROV data model. W3C Recommendation, 2013.

[8]. Edge D., Trinh H., Cheng N., Bradley J., Chao A., Mody A., Truitt S., Metropolitansky D., Osazuwa Ness R., Larson J. From local to global: A graph RAG approach to query-focused summarization. arXiv preprint, 2024. DOI: 10.48550/arXiv.2404.16130.

[9]. Peng B., Zhu Y., Liu Y., Bo X., Shi H., Hong C., Zhang Y., Tang S. Graph retrieval-augmented generation: A survey.

[10]. Lewis P., Perez E., Piktus A., Petroni F., Karpukhin V., Goyal N., Küttler H., Lewis M., Yih W., Rocktäschel T., Riedel S., Kiela D. Retrieval-augmented generation for knowledge-intensive NLP tasks. In Advances in Neural Information Processing Systems, 2020, vol. 33, pp. 9459-9474.

[11]. Yao S., Zhao J., Yu D., Du N., Shafran I., Narasimhan K.R., Cao Y. ReAct: Synergizing reasoning and acting in language models. In The Eleventh International Conference on Learning Representations, 2023. Thebibliography.

[12]. Wu Q., Bansal G., Zhang J., Wu Y., Li B., Zhu E., Jiang L., Zhang X., Zhang S., Liu J., Awadallah A.H., White R.W., Burger D., Wang C. AutoGen: Enabling next-gen LLM applications via multi-agent conversation. arXiv preprint, 2023. DOI: 10.48550/arXiv.2308.08155.

[13]. Guo T., Chen X., Wang Y., Chang R., Pei S., Chawla N.V., Wiest O., Zhang X.. Large language model based multi-agents: A survey of progress and challenges. In Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, 2024, pp. 8048-8057.

[14]. Belyaeva O., Bogatenkova A., Turdakov D. Dedoc: A universal system for extracting content and logical structure from textual documents. In 2023 Ivannikov ISPRAS Open Conference, 2023, pp. 20-25.

[15]. Реализация модели TDM. Доступно по адресу: https://github.com/ispras/tdm, дата обращения 10.08.2026.

Сведения об авторах / Information about authors

Денис Юрьевич ТУРДАКОВ – кандидат физико-математических наук, заведующий отделом информационных систем, заведующий исследовательским центром доверенного искусственного интеллекта Института системного программирования им. В.П. Иванникова РАН. Сфера научных интересов: машинное обучение, доверенный ИИ, извлечение информации, большие данные, анализ сложных сетей.

Denis Yuryevich TURDAKOV – Cand. Sci. (Phys.-Math.), Head of the Information Systems Department and Head of the Research Center for Trustworthy AI at the Ivannikov Institute for System Programming of the Russian Academy of Sciences. His research interests include machine learning, trustworthy artificial intelligence, information extraction, big data, and graph analysis.

Владимир Дмитриевич МАЙОРОВ – научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Основные научные интересы: обработка естественного языка, анализ тональности текстов, аспектно-ориентированный анализ мнений, большие языковые модели, агентные системы.

Vladimir Dmitrievich MAYOROV is a Researcher at the Ivannikov Institute for System Programming of the Russian Academy of Sciences. His research interests include natural language processing, sentiment analysis, aspect-based sentiment analysis, large language models, and agentic systems.

Максим Александрович РЫНДИН – научный сотрудник Института системного программирования им. В.П. Иванникова РАН. Научные интересы: MLOps, инфраструктура машинного обучения, автоматизация ML-конвейеров, контроль доступа и инженерия машинного обучения.

Maksim Alexandrovich RYNDIN is a Researcher at the Ivannikov Institute for System Programming of the Russian Academy of Sciences. His research interests include MLOps, machine learning infrastructure, automation of machine learning pipelines, access control, and machine learning engineering.