УДК 004.9
doi: 10.21685/2072-3059-2023-3-4

Представление и структурирование знаний в семантико-ориентированной вычислительной среде. Часть II. Интерпретации концептуальных событийных сетевых моделей для заданных предметных областей

В. И. Волчихин[1], Н. С. Карамышева[2], М. А. Митрохин[3], С. А. Зинкин[4]

[1,2,3,4]Пензенский государственный университет, Пенза, Россия

[1]cnit@pnzgu.ru, [2]karamyshevans@yandex.ru, [3]vt@pnzgu.ru, [4]zsa49@yandex.ru

Аннотация. *Актуальность и цели.* На основе предложенной методики глубокого структурирования знаний в семантико-ориентированной интеллектуальной вычислительной среде на основе расширения описательных возможностей сетей Петри путем их интеграции с концептуальными графами дана методика интерпретации концептуальных событийных сетевых моделей для заданных предметных областей. Под вычислительной средой понимается виртуальная распределенная вычислительная система, реализуемая на глобальной компьютерной сети. Подобно «Семантической паутине» (Semantic Web), в качестве основы для представления интеллектуальной информации о предметной области выбран концептуальный граф, интегрированный с логической сетью Петри в рамках одного формализма, пригодного для последующей машинной обработки. Схематическое представление концептуальных графов скрывает большую часть сложности, связанной с исчислением предикатов. Представленные примеры концептуальных графов с концептами-событиями содержат не только декларативную, но и процедурную составляющую модели представления знаний. Интеграция концептуальных графов с логическими сетями Петри, рассматриваемыми как особая разновидность семантических сетей, иллюстрируется рядом примеров из различных предметных областей. Описано также программное обеспечение, за счет которого реализуется данная интеграция в рамках одной модели представления знаний. Показано, что предметные области обусловливают структуризацию самой информатики и ее направлений развития, что относится и к интеллектуальным системам в частности. Целью работы является автоматизация выделения фактов и правил вывода для последующей программной реализации данного подхода в интеллектуальных событийных системах на примере конкретных предметных областей человеческой деятельности. *Материалы и методы*. Методологическая основа исследования предметной области ориентирована на использование имитационного моделирования интеллектуальных событийных систем, в которых взаимодействия компонентов задаются локально. В общем случае построение имитационной модели интеллектуальной событийной системы основано на анализе причинно-следственных ситуационных связей и правилах модификации как сигнатуры, так и конкретных предикатов и функций. *Результаты*. Реализована и проиллюстрирована примерами методика синтеза концептуальных логических сетей Петри на основе выявления общей семантики концептуальных графов и сетей Петри, в результате чего построены модели, обладающие декларативными, императивными и динамическими свойствами.

Ключевые слова: отношения, предикаты, концепты, продукционные правила, концептуальные графы, логические сети Петри, интеграция, интерпретация, структуризация и представление знаний, декларативность и императивность моделей, концептуальные сети Петри, динамика моделей

Для цитирования: Волчихин В. И., Карамышева Н. С., Митрохин М. А., Зинкин С. А. Представление и структурирование знаний в семантико-ориентированной вычислительной среде. Часть II. Интерпретации концептуальных событийных сетевых моделей для заданных предметных областей // Известия высших учебных заведений. Поволжский регион. Технические науки. 2023. № 3. С. 41–71. doi: 10.21685/2072-3059-2023-3-4

Representation and structuring of knowledge in the semantic oriented computing environment. Part 2. Interpretations of conceptual event network models for given subject areas

V.I. Volchikhin[1], N.S. Karamysheva[2], M.A. Mitrokhin[3], S.A. Zinkin[4]

[1,2,3,4]Penza State University, Penza, Russia

[1]cnit@pnzgu.ru, [2]karamyshevans@yandex.ru, [3]vt@pnzgu.ru, [4]zsa49@yandex.ru

Abstract. *Background.* Based on the proposed methodology for deep structuring of knowledge in a semantically-oriented intelligent computing environment based on expanding the descriptive capabilities of Petri nets by integrating them with conceptual graphs, a technique for interpreting conceptual event network models for given subject areas is given. The computing environment is understood as a virtual distributed computing system implemented on a global computer network. Like the "Semantic Web", a conceptual graph integrated with a logical Petri net within a single formalism suitable for subsequent machine processing is chosen as the basis for representing intellectual information about a subject area. The schematic representation of conceptual graphs hides much of the complexity associated with predicate calculus. The presented examples of conceptual graphs with event concepts contain not only a declarative, but also a procedural component of the knowledge representation model. The integration of conceptual graphs with logical Petri nets, considered as a number of examples from various subject areas illustrates a special type of semantic networks. The software through which this integration is implemented within the framework of one knowledge representation model is also described. It is shown that subject areas determine the structuring of computer science itself and its directions of development, which also applies to intelligent systems in particular. The purpose of the study is to automate the selection of facts and inference rules for the subsequent software implementation of this approach in intelligent event systems using the example of specific subject areas of human activity. *Materials and methods*. The methodological basis for researching the subject area is focused on the use of simulation modeling of intelligent event systems, in which the interactions of components are specified locally. In the general case, the construction of a simulation model of an intelligent event system is based on the analysis of cause-and-effect situational relationships and the rules for modifying both the signature and specific predicates and functions. *Results*. A method for synthesizing conceptual logical Petri nets are implemented and illustrated with examples based on identifying the general semantics of conceptual graphs and Petri nets, resulting in the construction of models with declarative, imperative and dynamic properties.

Keywords: relations, predicates, concepts, production rules, conceptual graphs, logical Petri nets, integration, interpretation, structuring and representation of knowledge, declarative and imperative models, conceptual Petri nets, model dynamics

For citation: Volchikhin V.I., Karamysheva N.S., Mitrokhin M.A., Zinkin S.A. Representation and structuring of knowledge in the semantic oriented computing environment. Part 2. Interpretations of conceptual event network models for given subject areas. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University pro-*

ceedings. Volga region. Engineering sciences. 2023;(3):41–71. (In Russ.). doi: 10.21685/2072-3059-2023-3-4

Введение

Современные приложения интеллектуальных систем исключительно разнообразны. Интеллектуальные системы являются составной частью информатики, которая, по одному из распространенных определений, является наукой о рациональной, преимущественно автоматической, обработке информации для поддержки человеческих знаний и коммуникаций в технической, экономической и социальной областях [1]. Развитие информатики и как научной, и как прикладной дисциплины связано с тем, что ее методы востребованы практически во всех предметных областях науки, образования и промышленности, поэтому могут использоваться практически во всех сферах научного познания, привнося в них принципиально новые качества. Но стоит заметить, что предметные области обусловливают структуризацию самой информатики и ее направлений развития, что относится и к интеллектуальным системам в частности.

Очевидно, что выбор средств для манипуляции с предметной областью зависит от ее семантики. Известен ряд языков для описания семантики предметной области. Основой для описания различных предметных областей при создании интеллектуальных систем может быть, например, «Семантическая паутина» – Semantic Web, являющаяся надстройкой над существующей «Всемирной паутиной» – World Wide Web, которая призвана сделать размещенную в ней информацию более понятной для компьютеров [2]. Особенностью предложенных принципов построения Semantic Web является повсеместное использование унифицированных идентификаторов – URI (*англ. Uniform Resource Identifier*), которые представляют собой последовательность символов, идентифицирующих абстрактный или физический ресурс. Свои глобально уникальные URI в Semantic Web есть не только у веб-страниц, но и у объектов реального мира – людей, городов, художественных произведений, а также у свойств, например таких, как «имя», «должность», «цвет». Программа-клиент может непосредственно извлекать из «паутины» факты и делать из них логические заключения [3].

В работе [4] показано, что исследования и разработки в области создания семантических моделей проектирования программного обеспечения развиваются в настоящее время в рамках нового направления ODSD (англ. *Ontology-Driven Software Engineering*). Показательным примером работы в данном направлении является европейский проект MOST (англ. *Marrying Ontology and Software Technology*) [5], в котором реализована «бесшовная» интеграция семантических технологий в разработку программного обеспечения на основе моделей MDSD (англ. *Model Driven Software Development*).

В настоящей работе рассмотрен подход к формализации концептуальных графов как декларативного, так и императивного (процедурного) характера, интегрированных с сетями Петри. ***Целью работы*** является автоматизация представления фактов и правил вывода при последующей программной реализации данного подхода в интеллектуальных событийных системах на примере конкретных предметных областей человеческой деятельности. Подобно Semantic Web, основой для представления интеллектуальной информа-

ции о предметной области будет семантическая сеть или концептуальный граф, пригодные для последующей машинной обработки.

Определения модифицируемых концептуальных графов

Определение взаимодействий в интеллектуальных событийных системах

При интеграции концептуальных графов и сетей Петри в объединенную модель представления знаний о предметной области [6] появляется возможность учета многих видов отношений: причинно-следственных (каузальных), предшествования-следования, родовидовых, класс-подкласс, целое-часть, синонимии и антонимии, логических, функциональных, атрибутивных, количественных, пространственных, временных, лингвистических, характерных для интеллектуальных систем. Обширный список отношений представлен, например, в работе [7], посвященной ситуационному управлению в сложных системах.

Следуя работам [8–12], рассмотрим методологическую основу исследования предметной области, ориентированной на использование имитационного моделирования интеллектуальных событийных систем, в которых взаимодействия компонентов задаются локально. В общем случае построение имитационной модели интеллектуальной событийной системы основано на анализе причинно-следственных ситуационных связей, заданных следующим отношением:

$$Rel \subseteq \mathbf{P}(S) \times \mathbf{P}(S), \tag{1}$$

где $\mathbf{P}$ – символ булеана; S – множество ситуаций.

Определена интерпретация сигнатуры:

$$I\colon \Sigma \to S, \tag{2}$$

где Σ – сигнатура (множество функциональных и предикатных символов конечных арностей) с интерпретацией I в множестве ситуаций S, т.е. во множестве конкретных функций и предикатов. Взаимодействия компонентов в моделируемой событийной системе задаются локально (кроме того, при независимых локальных ситуациях акты взаимодействия компонентов могут выполняться параллельно) на основе выполнения правил модификации интерпретации следующего вида:

$$I(\sigma_i) \leftarrow s_i, \tag{3}$$

где $\sigma_i \in \Sigma$ – функциональный или предикатный символ; $s_i \in S$ – текущая локальная ситуация в предметной области, определяемая конкретной функцией или предикатом.

В многосортных, или многоосновных, системах тип n-арного функционального символа – это кортеж $(i_1, i_2, \ldots, i_n, j)$, а тип n-арного предикатного символа – это кортеж $(i_1, i_2, \ldots, i_n)$, где $i_1, i_2, \ldots, i_n, j$ – названия (сорта) для основ или носителей [13, 14]. При определении модифицируемых концептуальных графов используются правила модификации их структуры при помощи правил вывода. Модификация конкретного многосортного предиката или функции в общем случае выполняется в соответствии со следующим правилом вывода:

$$\frac{t_1, t_2, \ldots, t_k, t_{k+1}}{s(t_1, t_2, \ldots, t_k) \leftarrow t_{k+1}}, \quad (4)$$

где t_1, t_2, …, t_k – термы различных сортов (при определении концептуальных графов термами чаще всего являются предметные переменные или предметные константы); s – функциональный или предикатный символ; "←" – символ операции присваивания. В случае, если s – функциональный символ, t_{k+1} – суть терм любого сорта, а если s – предикатный символ, то t_{k+1} должен быть булевым термом.

Методы построения концептуальных графов на основе n-арных ($n \geq 1$) предикатов были рассмотрены в работе [6] на конкретном примере, следуя известной методике, описанной в работе [15]. В связи с тем, что концептуальные графы и семантические сети обычно строятся на базе унарных и бинарных предикатов, правило модификации бинарного предиката будет использоваться в следующем частном виде:

$$\frac{t_1, t_2, t_3}{p_1(t_1, t_2) \leftarrow t_3}, \quad (5)$$

а правило модификации унарного предиката – в следующем частном виде:

$$\frac{t_1, t_3}{p_2(t_1) \leftarrow t_3}. \quad (6)$$

В правилах (5) и (6) t_1, t_2 – термы различных сортов, p_1 – бинарный предикатный символ, p_2 – унарный предикатный символ, t_3 – булев терм.

Правило модификации унарной функции имеет следующий вид:

$$\frac{t_1, t_3}{f(t_1) \leftarrow t_3}, \quad (7)$$

где t_1, t_3 – термы различных сортов, f – унарный функциональный символ.

Выражениями (1)–(7) определяются модифицируемые взаимодействия в интеллектуальных событийных системах. Далее на данной основе возможно определить важные для приложений модифицируемые концептуальные графы.

Формальное определение простых и модифицируемых концептуальных графов

Обычный концептуальный граф представляет собой размеченный двудольный ориентированный граф:

$$CG_1 = (Pred, Arg, U), \quad (8)$$

где $Arg \cap Pred = \varnothing$, вершины из множества $Pred$ размечены именами предикатов, а вершины из множества Arg – именами аргументов; $U \subseteq (Pred \times Arg) \cup (Arg \times Pred)$ – отношение инцидентности, определяющее множество дуг. Размеченные вершины из множества $Pred$ называются концептуальными предикатами, а размеченные вершины из множества Arg – вершинами-концептами, или просто концептами [16–18]. Предикатам соот-

ветствуют одноименные отношения (по определению, отношение является областью истинности предиката). По определению дуг, они могут связывать предикатные вершины с вершинами-концептами и наоборот; не допускается непосредственное связывание вершин из одних и тех же множеств *Pred* и *Arg*. Вместо приведенного определения концептуальных графов вида CG_1 чаще используются их формульные описания в рамках логики высказываний и первопорядковой логики предикатов.

При учете того факта, что часть отношений может иметь функциональный характер, следует расширить определение (8):

$$CG_2 = (Pred,\ Arg, Func, U_p, U_f), \tag{9}$$

где $Arg \cap Pred = \varnothing$, $U_p \subseteq (Pred \times Arg) \cup (Arg \times Pred)$ – отношение инцидентности, определяющее множество дуг, связывающих предикатные вершины с аргументами-концептами; $Pred \cap Func = \varnothing$, $Arg \cap Func = \varnothing$, $U_f \subseteq (Func \times Arg) \cup (Arg \times Func)$ – отношение инцидентности, определяющее дуги, связывающие функциональные вершины с аргументами-концептами из области определения функций с областью их значений.

Модифицируемый концептуальный граф определяется следующим кортежем:

$$CG_3 = (Pred,\ Arg, Func, U_p, U_f, V_p, V_f), \tag{10}$$

где дополнительно по сравнению с кортежем (9) определены два множества V_p и V_f правил модификации предикатов и функций соответственно, поэтому структура концептуального графа может изменяться при моделировании.

Формальное определение последовательности концептуальных графов, задаваемых в процессе моделирования или проектирования интеллектуальной системы

На основе кортежа (10) определяется последовательность концептуальных графов, предназначенных для поведенческого моделирования и проектирования сложных программных систем. Рассматривается множество сигнатур вида $\Sigma_k = (Pred_k, Func_k)$, $k = 1, 2,\ldots, n$, где k – номер этапа моделирования или проектирования интеллектуальной системы, во время которого сигнатура Σ_k остается постоянной; $Pred_k$ – множество унарных и бинарных предикатных символов; $Func_k$ – множество унарных предикатных символов. При переходе от одной сигнатуры к другой каждая последующая сигнатура может обогащаться или обедняться соответственно за счет введения или удаления некоторых символов из множеств $Pred_k$ и $Func_k$. На каждом k-м этапе интерпретация I_k сигнатуры Σ_k может изменяться за счет согласованного применения правил модификации из множеств V_p и V_f к конкретным предикатам и функциям.

Формальное определение иерархических концептуальных графов с модифицируемой структурой

Структурированный, или иерархический, концептуальный граф определяется следующим кортежем:

$$CG_4 = (Pred,\ Arg, Func, U_p, U_f, V_p, V_f, N), \tag{11}$$

где N – множество концептов-высказываний, представляющих собой концептуальные подграфы (возможно, вложенные друг в друга), описываемые формулами в логике предикатов. Примеры подобного структурирования и вложения концептов-высказываний представлены в работе [6].

Другое понятие об иерархии в концептуальных графах связано с наличием в них иерархий классов с отношениями типа «элемент класса – класс» (отношения «*конкретизации*», или "*is-a*") и с отношениями типа «подкласс – класс» (отношения «*это*», или "*a kind of*").

Определение процедурной составляющей в модели представления знаний на основе концептуального графа

Кортежами (8)–(11) определены в основном декларативные свойства моделей представления знаний концептуальными графами. Например, модифицируемый концептуальный граф CG_3 определен кортежем (10), в котором процедурная составляющая знаний представлена неявно, при помощи двух множеств V_p и V_f, правил модификации предикатов и функций, которые реализуются на основе выражений (3)–(7). Правило (3) задает закрепление конкретного предиката или функции за предикатным или функциональным символом соответственно. Действия по модификации конкретных предикатов и функций выполняются при помощи правил (4)–(7). Данные правила целесообразно объединить в группы совместимых составных правил, образуя абстрактные модули (АМ), реализуемые впоследствии программно.

Процедурную составляющую модели представления знаний можно задать одним из следующих способов:

1) непосредственно в концептуальном графе в множество размеченных вершин-концептов *Arg* включить подмножество концептов-событий $Event \subseteq Arg$, связывая их отношениями (предикатными вершинами) предшествования-следования *Pred* или *Next* и вводя дополнительные условия, определяя алгоритм работы АМ. На этой основе возможно задавать все структурированные дейкстровские конструкции для будущих программ на конкретных алгоритмических языках;

2) возможно определить АМ при помощи суперпозиции операций системы алгоритмических алгебр (САА) Глушкова [19], связывающих между собой концепты-события из множества *Event*;

3) учитывая тот факт, что ингибиторная сеть Петри является универсальной алгоритмической системой [20–22], АМ возможно определить при помощи согласованных правил срабатывания переходов подобной сети, вводя дополнительные предикаты, моделирующие условные переходы в программах. Формальное определение параллельной программы на основе сети Петри, обладающей свойствами свободного выбора, живости и безопасности, приведено в работе [22]. В определение включены множество предикатов и функция распределения предикатов по множеству позиций, обладающих свойством свободного выбора, т.е. таких позиций, которые являются единственными входными позициями для нескольких переходов.

Пример интерпретации концептуального графа «Почта»

На рис. 1 приведен пример концептуального графа с контекстным представлением для фразы, описывающей отправку письма по электронной почте.

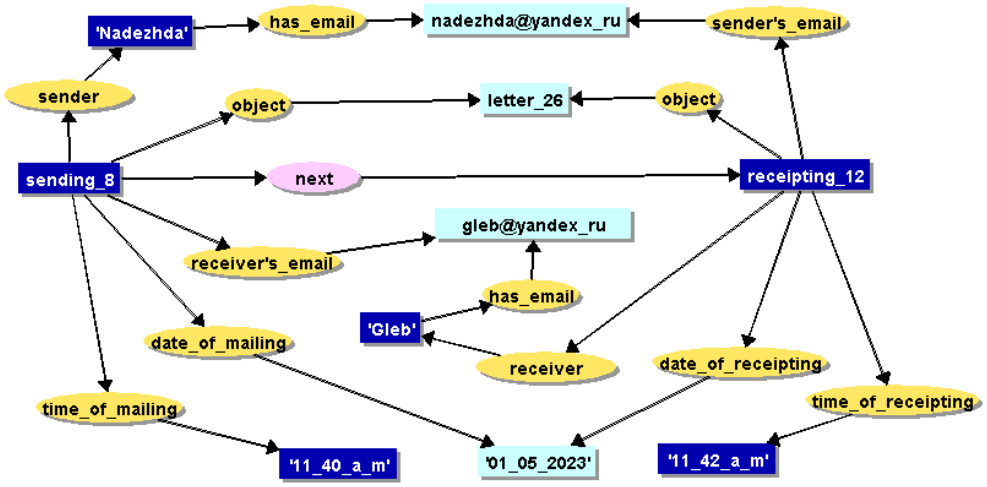


Рис. 1. Пример концептуального графа G_1 с контекстным представлением

Данный граф получен из двух концептуальных подграфов G_{11} и G_{12}, соединенных по правилу конъюнкции [15] концептов *nadezhda@yandex.ru*, *letter_26*, *gleb@yandex.ru*, '*01_05_2023*', принадлежащих как одному, так и другому подграфу. Подграф G_{11} описывает событие-концепт *sending_8* отправки письма по электронной почте, а подграф G_{12} – событие-концепт *receipting_12* приема этого письма. В концептуальном графе G_1 событие-концепт *sending_8* связано с событием-концептом *receipting_12* отношением *next*. В поведенческой модели эта связь активизируется путем модификации бинарного предиката:

$$next(sending\_8, receipting\_12) \leftarrow \textbf{true},$$

в результате выполнения которого становится истинным высказывание *next*(*sending_8*, *receipting_12*). Это простейший случай применения правила (5).

Редактор концептуальных графов *CharGer* [23, 24] позволяет по XML-коду графа G_1 получить информацию, необходимую для создания программного обеспечения интеллектуальной системы. Высказывания, приведенные ниже, получены при помощи специальной программы-переходника и полностью готовы для использования в качестве фактов для базы знаний, создаваемой на языке Prolog:

```
Facts in a Prolog program:
date_of_mailing(sending_8, '01_05_2023').
date_of_receipting(receipting_12, '01_05_2023').
has_email('Gleb', gleb_yandex_ru).
has_email('Nadezhda', nadezhda_yandex_ru).
next(sending_8, receipting_12).
object(receipting_12, letter_26).
object(sending_8, letter_26).
receiver's_email(sending_8, gleb_yandex_ru).
receiver(receipting_12, 'Gleb').
sender's_email(receipting_12, nadezhda_yandex_ru).
sender(sending_8, 'Nadezhda').
time_of_mailing(sending_8, '11_40_a_m').
time_of_receipting(receipting_12, '11_42_a_m').
```

Приведенная методика позволяет автоматизировать наиболее трудоемкий этап в создании интеллектуальных систем, а именно этап формирования базы фактов на основе концептуального графа. Ее недостатком является то, что верификация концептуальных графов распределенных алгоритмов не предусмотрена в существующих реализациях интеллектуальных редакторов концептуальных графов.

Формализация и последующая реализация третьего способа является предпочтительной, поскольку как для концептуальных графов, так и для сетей Петри имеются интеллектуальные редакторы и средства анализа. При интеграции этих двух моделей появится возможность создания интеллектуальных систем, где бо́льшую роль будут играть концепты-события, роль которых в качестве элементов процедурной составляющей модели представления знаний будет усилена за счет сетей Петри. Сети Петри обогащаются за счет возможности указания свойств объектов, представленных переходами и позициями.

Определение и моделирование интеллектуальных систем на основе концептуальных логических сетей Петри

Акторы – переходы в концептуальных логических сетях Петри

Сети Петри могут рассматриваться как семантические сети с событиями частного вида, или как сценарии деятельности, предназначенные для исследования статических и динамических свойств различных систем: систем отношений между людьми, последовательностей действий при выполнении какой-либо работы, последовательностей передачи сообщений в сетях ЭВМ и др.

В логической, или бинарной, сети Петри позициям соответствуют условия, которые представляются простыми или составными высказываниями, а переходы объединяют высказывания чаще всего в выражениях-конъюнкциях, истинность которых свидетельствует о готовности срабатывания переходов. При исследовании систем на основе логических сетей Петри могут быть построены как логические модели на основе декларативных языков программирования, так и продукционные модели, которые имеют не только декларативный, но и процедурный, или императивный, характер, что облегчает создание приложений на универсальных процедурных и объектно-ориентированных языках.

В отличие от концептуальных сетей Петри, определенных или использованных в работах [6, 25–27], в предлагаемых в настоящей работе концептуальных логических сетях Петри (КЛ СП) для представления переходов используются так называемые *акторы*, применяемые в редакторе концептуальных графов в качестве вершин для реализации логических и арифметических функций. Новая семантика акторов-переходов предопределена таким образом, что они способны в соответствии с новой интерпретацией выполнять функции переходов сети Петри, работая в соответствии с логическими правилами срабатывания переходов.

Определяемая далее логика событий, происходящих на входах и выходах акторов-переходов, в целом соответствует известному методу графической оценки и анализа при сетевом планировании GERT (англ. *Graphical Evaluation and Review Technique*), отдельные модификации которого нашли

свое применение при анализе конкурирующих и параллельных программ для ЭВМ [28]. Основным методом исследования сложных систем с применением сетей GERT и Q-GERT (модель сети GERT с очередями) является метод статистического моделирования – метод Монте-Карло. Аналитические методы исследования сетей сложны и практически нереализуемы.

Определение логической и концептуальной сетей Петри с переменной структурой

Логическая, или бинарная, сеть Петри *LPN* (англ. *Logical Petry Net*) может быть задана следующим кортежем:

$$LPN = (A, T, R, UR), \tag{12}$$

где A – множество позиций, интерпретируемых как предметные константы; T – множество переходов, состоящее из двух непересекающихся подмножеств $T^{\&}$ и T^*, где $T = T^{\&} \cup T^*$, $T^{\&} \cap T^* = \varnothing$, $T_{\&}$ – подмножество переходов с конъюнктивной логикой на входах и выходах, T^* – подмножество переходов с конъюнктивной логикой на входах и с недетерминированным или условным выбором одного из логических правил вывода (правил модификации предиката разметки позиций $R(a)$, $a \in A$); если $T^* = \varnothing$, то допускается использование обозначения T для переходов типа $T^{\&}$; R: $A \to \{\textbf{true}, \textbf{false}\}$ – унарный предикат разметки позиций, модифицируемый после каждого срабатывания какого-либо перехода; начальная интерпретация предиката $R(a)$, $a \in A$, соответствует начальной разметке позиций сети *LPN*; *UR* – множество правил модификации (англ. *Update Rules*) унарного предиката $R(a)$, $a \in A$, в процессе срабатывания переходов; правила *UR* в неявной форме задают также и структуру связей в сети Петри типа *LPN*.

Концептуальная логическая сеть Петри КЛ СП, или *CLPN* (англ. *Conceptual Logical Petry Net*), с переменной структурой задается на основе кортежа (12) определения сети *LPN* следующим образом:

$$CLPN = (A, UP, MUP, BP, NBP, T, R, UR), \tag{13}$$

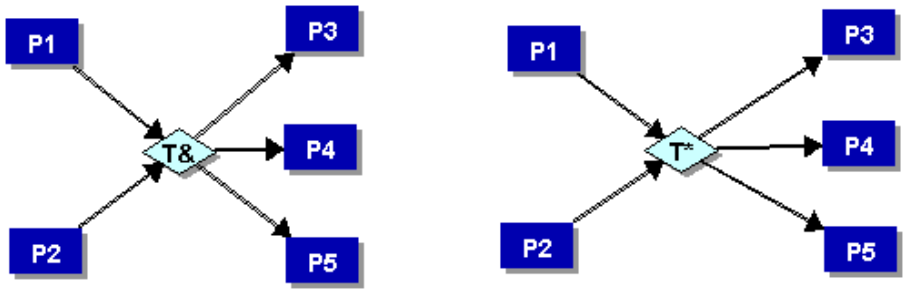
где A – множество концептов-позиций, интерпретируемых как предметные константы; *UP* – множество унарных предикатов для обозначения свойств концептов-позиций; *MUP* – множество правил модификации свойств концептов-позиций; *BP* – множество бинарных предикатов для обозначения отношений между концептами-позициями; *NBP* – множество правил модификации отношений между концептами-позициями; T – множество концептов-переходов, состоящее из двух непересекающихся подмножеств $T^{\&}$ и T^*, где $T = T^{\&} \cup T^*$, $T^{\&} \cap T^* = \varnothing$, $T^{\&}$ – подмножество концептов-переходов с конъюнктивной логикой на входах и выходах, T^* – подмножество концептов-переходов с конъюнктивной логикой на входах и с недетерминированным или условным выбором одного из логических правил вывода (правил модификации предиката разметки позиций $R(a)$, $a \in A$); если $T^* = \varnothing$, то допускается использование обозначения T для переходов типа $T^{\&}$; R: $A \to \{\textbf{true}, \textbf{false}\}$ – унарный предикат разметки концептов-позиций, модифицируемый после каждого срабатывания какого-либо концепта-перехода; начальная интерпретация предиката $R(a)$, $a \in A$, соответствует начальной

разметке концептов-позиций сети *CLPN*; *UR* – множество правил модификации (англ. *Update Rules*) унарного предиката $R(a)$, $a \in A$, в процессе срабатывания переходов; правила *UR* в неявной форме задают также и структуру связей в сети Петри типа *CLPN*.

Элементы кортежа *UP*, *MUP*, *BP* и *NBP* при их интерпретации задают правила модификации концептуального графа, положенного в основу сети *CLPN*.

Методика использования акторов-переходов в концептуальных логических сетях Петри

Использование акторов-переходов иллюстрирует рис. 2. Количество входных (2) и выходных позиций (3) для данного примера определено из соображений компактности графов и соответствующих им правил. Считается, что позициям соответствуют одноименные им переменные высказывания.



а) ***б***)

Рис. 2. Акторы-переходы $T^{\&}$ (***а***) и T^* (***б***)
в концептуальных логических сетях Петри (КЛ СП)

По умолчанию на входе и на выходе актора-перехода $T^{\&}$ (рис. 2,***а***) реализуются операции конъюнкции. На входе актора-перехода T^* реализуется операция конъюнкции; опционально (вручную в редакторе) на выходе актора T^* (рис. 2,***б***) может быть задана реализация операции выбора одной из нескольких альтернатив по условиям, недетерминированно или по вероятностям.

Выход по условиям соответствует выполнению управляющих операторов *switch* или *case*, применяемых в языках программирования. Под недетерминированным выбором подразумевается выбор одного из выходных высказываний «внешним, или независимым, наблюдателем». При вероятностном выборе, который также относится к недетерминированным, каждое исходящее действие, состоящее в появлении единственного истинного высказывания на выходе актора-перехода, имеет некоторую вероятность выполнения. Сумма вероятностей выходов должна быть равна единице.

Редактор концептуальных графов CharGer, обрабатывая внутренние XML/PNML-представления концептуальных графов на рис. 2, выдает следующие правила срабатывания переходов:

а) Impact Results $T^{\&}$ of P_1 and P_2 are P_3 and P_4 and P_5;
б) Impact Result T^* of P_1 and P_2 is P_3;
Impact Result T^* of P_1 and P_2 is P_4;
Impact Result T^* of P_1 and P_2 is P_5.

Они имеют следующий смысл:

– для случая ***а*** (рис. 2): результатом применения актора-перехода $T^{\&}$ к истинным высказываниям P_1 и P_2 станет истинность высказываний P_3, P_4 и P_5; предполагается, что до срабатывания актора-перехода высказывания P_3, P_4 и P_5 были ложными, что для безопасной сети Петри означает отсутствие меток в одноименных позициях; далее этот факт для логических сетей Петри будет подразумеваться по умолчанию;

– для случая ***б*** (рис. 2) в результате проверки введенных условий или недетерминированно, или по заданным вероятностям выбирается одно из трех правил. Например, первое правило срабатывания перехода имеет следующий смысл: результатом применения актора-перехода T^* к истинным высказываниям P_1 и P_2 станет истинность высказывания P_3. Правила срабатывания переходов для случаев, когда позициям соответствуют элементарные высказывания, имеют следующий вид:

$$\textbf{\textit{а}})\ T^{\&}: P_1 \,\&\, P_2 \to P_3 \,\&\, P_4 \,\&\, P_5; \tag{14}$$

$$\textbf{\textit{б}})\ T^*: P_1 \,\&\, P_2 \to P_3; \tag{15}$$

или

$$T^*: P_1 \,\&\, P_2 \to P_4;$$

или

$$T^*: P_1 \,\&\, P_2 \to P_5.$$

Скорректированные правила (16) и (17) содержат конъюнкции $\neg P_1 \,\&\, \neg P_2$ в правых частях, учитывающие перемещение меток через переходы:

$$\textbf{\textit{а}})\ T^{\&}: P_1 \,\&\, P_2 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_3 \,\&\, P_4 \,\&\, P_5; \tag{16}$$

$$\textbf{\textit{б}})\ T^*: P_1 \,\&\, P_2 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_3; \tag{17}$$

или

$$T^*: P_1 \,\&\, P_2 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_4;$$

или

$$T^*: P_1 \,\&\, P_2 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_5.$$

Следующие правила (18) и (19) отличаются от предыдущих добавлением конъюнкций в левых частях, что соответствует проверкам незанятости выходных позиций:

$$\textbf{\textit{а}})\ T^{\&}: P_1 \,\&\, P_2 \,\&\, \neg P_3 \,\&\, \neg P_4 \,\&\, \neg P_5 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_3 \,\&\, P_4 \,\&\, P_5; \tag{18}$$

$$\textbf{\textit{б}})\ T^*: P_1 \,\&\, P_2 \,\&\, \neg P_3 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_3; \tag{19}$$

или

$$T^*: P_1 \,\&\, P_2 \,\&\, \neg P_4 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_4;$$

или

$$T^*: P_1 \,\&\, P_2 \,\&\, \neg P_5 \to \neg P_1 \,\&\, \neg P_2 \,\&\, P_5.$$

Использование элементарных высказываний для одноименных позиций в формулах (14)–(19) упрощает составление программ, интерпретирующих КЛ СП.

Следующие выражения (20) и (21) содержат высказывания, записанные в соответствии с определениями (12) и (13) при помощи предиката R разметки позиций:

а) $T^{\&}$: $R(P_1)\ \&\ R(P_2) \to \neg\, R(P_1)\ \&\ \neg\, R(P_2)\ \&\ R(P_3)\ \&\ R(P_4)\ \&\ R(P_5)$; (20)

б) T^{*}: $R(P_1)\ \&\ R(P_2) \to \neg\, R(P_1)\ \&\ \neg\, R(P_2)\ \&\ R(P_3)$; (21)

или

T^{*}: $R(P_1)\ \&\ R(P_2) \to \neg\, R(P_1)\ \&\ \neg\, R(\mathrm{P}_2)\ \&\ R(P_4)$;

или

T^{*}: $R(P_1)\ \&\ R(P_2) \to \neg\, R(P_1)\ \&\ \neg\, R(P_2)\ \&\ R(P_5)$.

Следующие выражения для правил срабатывания переходов будут записаны на основе формул (20 и (21) в виде α-дизъюнкций из операций системы алгоритмических алгебр Глушкова; условия α, при истинности которых происходит срабатывание переходов, заключены в квадратные скобки, а в круглые скобки слева от символа неравнозначности "⊕" заключены одна или несколько согласованных операций модификации предиката R разметки позиций; если α-условие ложно, то переход не срабатывает, что соответствует выполнению пустого оператора E; обратной стрелкой "←" обозначена операция модификации предиката R:

а) $T^{\&}$: $[R(P_1)\ \&\ R(P_2)](R(P_1) \leftarrow R(P_1), R(P_2) \leftarrow \neg\, R(P_2), R(P_3) \leftarrow \mathbf{true}$, $R(P_4) \leftarrow \mathbf{true}, R(P_5) \leftarrow \mathbf{true} \oplus E$); (22)

б) T^{*}: $[R(P_1)\ \&\ R(P_2)](R(P_1) \leftarrow \neg\, R(P_1), R(P_2) \leftarrow \neg\, R(P_2),$ (23)
$R(P_3) \leftarrow \mathbf{true} \oplus E$);

или

T^{*}: $[R(P_1)\ \&\ R(P_2)](R(P_1) \leftarrow \neg\, R(P_1), R(P_2) \leftarrow \neg\, R(P_2),$
$R(P_4) \leftarrow \mathbf{true} \oplus E$);

или

T^{*}: $[R(P_1)\ \&\ R(P_2)](R(P_1) \leftarrow \neg\, R(P_1), R(P_2) \leftarrow \neg\, R(P_2),$
$R(P_5) \leftarrow \mathbf{true} \oplus E$).

Позиции в КЛ СП рассматриваются как обычные концепты и могут иметь окружение в виде отношений и других концептов, как показано на рис. 3, где овалами графически представлены бинарные отношения между концептами. Например, при функциональном характере отношений они могут указывать роли для объектов.

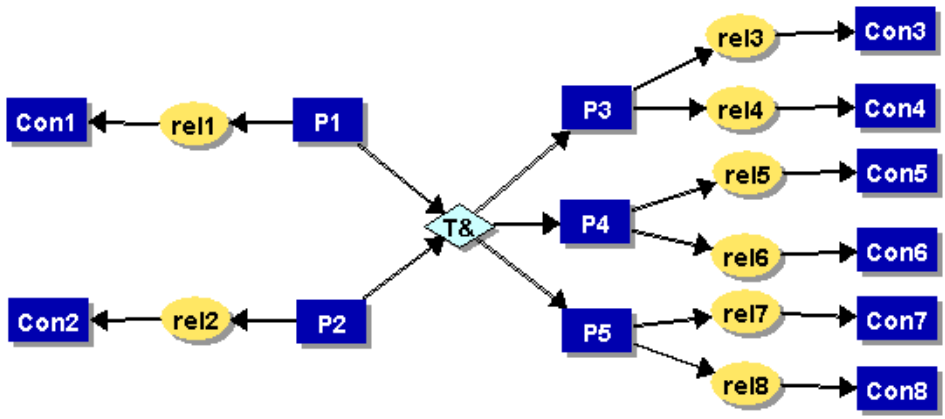


Рис. 3. Актор-переход в составе фрагмента концептуальной логической сети Петри

Расширенному набору операций на входах и выходах акторов-переходов КЛ СП ставятся в соответствие следующие обозначения:

$$T^{\&\&}, (T^{\&}, T); T^{\&}*, (T*); T*^{\&}; T**,$$

где первые символы "&" и "*" в верхних индексах обозначают входную логику, определенную ранее. Вторые индексы соответствуют подобной же логике, только реализуемой на выходе акторов-переходов. Рядом в скобках указаны упрощенные эквивалентные обозначения для акторов-переходов.

Пример построения предметно интерпретируемой концептуальной логической сети Петри для правового процесса

Моделирование элементов правового процесса

Уже на начальных этапах активного исследования сети Петри и их разновидности использовались при моделировании большого числа систем, в том числе и отличных от вычислительных систем. Например, в основополагающей монографии [20] предлагался перечень задач на применение сетей Петри в следующих предметных областях: исследование операций, моделирование мозга, химические реакции, военный конфликт, политические системы, экономика и макроэкономические события, транспортные потоки на дорогах, биологические популяции, семантические сети для представления естественного языка, языкознание и литературоведение.

В качестве примера рассматривается гипотетический процесс подачи и продвижения условного искового заявления в юридической практике [20, 29]. Это простой процесс передачи и регистрации документов, характерный и для других областей человеческой деятельности. Процесс описывается концептуальной логической сетью Петри, приведенной на рис. 4.

В правовых системах несколько действующих лиц (судьи, адвокаты, обвиняемые, клерки и др.) могут одновременно или последовательно выполнять действия, относящиеся к конкретному делу. На рис. 4, в отличие от традиционных обозначений элементов обычных сетей Петри, прямоугольники представляют позиции-концепты, а ромбы – акторы-переходы.

Подобное описание позволит в дальнейшем упростить исследование моделируемых процессов на основе логического подхода в искусственном интеллекте, когда на начальном этапе требуется исследовать данную сеть как обычную сеть Петри и показать, что эта сеть живая и безопасная (1-ограниченная), т.е. в каждой позиции может находиться не более одной метки. Если сеть Петри обладает свойством безопасности, то далее возможно построить на ее основе логическую и простую продукционную модели представления знаний о предметной области.

Как следует из комментариев на рис. 4, переходы в данной сети имеют следующую содержательную интерпретацию [20, 29]:

T_0 – ИСТЕЦ регистрирует иск у КЛЕРКА;

T_1 – КЛЕРК выписывает вызов в суд;

T_2 – КЛЕРК передает вызов и иск СУДЕБНОМУ ИСПОЛНИТЕЛЮ;

T_3 – СУДЕБНЫЙ ИСПОЛНИТЕЛЬ сообщает ОТВЕТЧИКУ об иске и вручает ему вызов в суд;

T_4 – ОТВЕТЧИК решает отвечать;

T_5 – ОТВЕТЧИК решает дать встречный иск;

T_6 – переход определяет, что время ожидания со дня передачи вызова в суд ОТВЕТЧИКУ истекло;

T_7 – ОТВЕТЧИК официально отвечает ИСТЦУ;

T_8 – ОТВЕТЧИК выставляет встречный иск;

T_9 – ОТВЕТЧИК не является в суд.

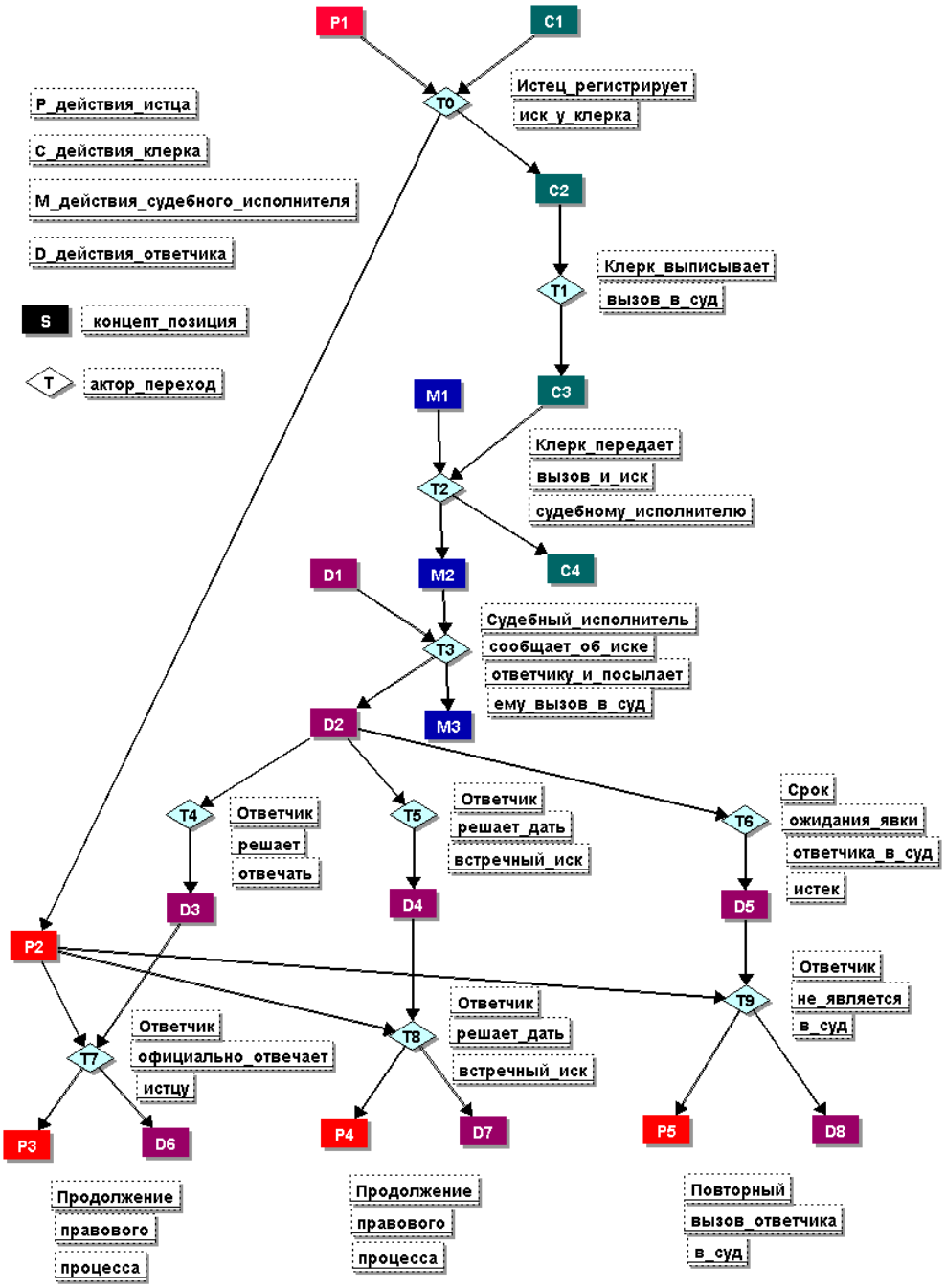


Рис. 4. Концептуальный граф $CLPN_1$ логической сети Петри, описывающей процесс подачи и рассмотрения искового заявления

Клерк в данном процессе выполняет функции секретаря суда или делопроизводителя, работающего в канцелярии суда. Позиции в рассматриваемой сети Петри соответствуют состояниям участников процесса. Последовательности прохождения меток по концептам-позициям и акторам-переходам соответствуют процессам ИСТЦА, КЛЕРКА, СУДЕБНОГО ИСПОЛНИТЕЛЯ и ОТВЕТЧИКА:

P – процесс ИСТЦА:

$P_1 \to T_0 \to P_2 \to (T_7 \to P_3 \mid T_8 \to P_4 \mid T_9 \to P_5)$;

C – процесс КЛЕРКА:

$C_1 \to T_0 \to C_2 \to T_2 \to C_4$;

M – процесс СУДЕБНОГО ИСПОЛНИТЕЛЯ:

$M_1 \to T_2 \to M_2 \to T_3 \to M_3$;

D – процесс ОТВЕТЧИКА:

$D_1 \to T_3 \to D_2 \to (T_4 \to D_3 \to T_7 \to D_6 \vee T_5 \to D_4 \to T_8 \to D_7 \vee T_6 \to \to D_5 \to T_9 \to D_8)$.

Символ "|" означает, что одна из последовательностей, заключенная в скобки, выбирается под воздействием внешних причин. Для процесса P ИСТЦА выбор одной из последовательностей $T_7 \to P_3 \mid T_8 \to P_4 \mid T_9 \to P_5$ зависит от поведения ОТВЕТЧИКА; для процесса D выбор одной из последовательностей $T_4 \to D_3 \to T_7 \to D_6 \vee T_5 \to D_4 \to T_8 \to D_7 \vee T_6 \to D_5 \to T_9 \to D_8$ зависит от поведения самого ОТВЕТЧИКА.

Определение концептуальной логической сети Петри типа CLPN для конкретной предметной области – правового процесса

Концептуальная логическая, или бинарная, сеть Петри для гипотетического правового процесса, представленная на рис. 4, формально задается следующим кортежем:

$$CLPN_1 = (A, T, R, UR),$$

где элементы A, T, R и UR определены непосредственно для процесса подачи и прохождения искового заявления (по сравнению с сетью *CLPN* общего вида элементы кортежа UP, MUP, BP и NBP в формуле (13) для данной задачи не определены):

$A = P \cup C \cup M \cup D$ – множество позиций, отмечающих состояния участников процесса;

$P = \{P_1, P_2, P_3, P_4, P_5\}$ – множество позиций, отмечающих состояния процесса ИСТЦА;

$C = \{C_1, C_2, C_3, C_4\}$ – множество позиций, отмечающих состояния процесса КЛЕРКА;

$M = \{M_1, M_2, M_3\}$ – множество позиций, отмечающих состояния процесса СУДЕБНОГО ИСПОЛНИТЕЛЯ;

$D = \{D_1, D_2, D_3, D_4, D_5, D_6, D_7, D_8\}$ – множество позиций, отмечающих состояния процесса ОТВЕТЧИКА;

$T = \{T_0, T_1, T_2, T_3, T_4, T_5, T_6, T_7, T_8, T_9\}$ – множество переходов-инициаторов действий участников процесса;

R: $A \to \{$**true**, **false**$\}$ – унарный предикат разметки позиций, модифицируемый после каждого срабатывания какого-либо перехода; начальная интерпретация предиката $R(a)$, $a \in A$, соответствует начальной разметке сети $CLPN_1$;

$UR = \{URT_0, URT_1, URT_2, URT_3, URT_4, URT_5, URT_6, URT_7, URT_8, URT_9\}$ – множество правил модификации предиката $R(a)$, $a \in A$, в процессе срабатывания переходов.

Выражения для правил UR модификации предиката R(a) разметки позиций в логической модели правового процесса

В результате автоматического логического анализа по внутреннему XML/PNML представлению концептуального графа логической сети Петри, изображенной на рис. 4, автоматически получена система правил, показанная на рис. 5.

There are the Propositions where:
Impact Results of T0 of C1 and P1 are P2 and C2
Impact Result of T1 of C2 is C3
Impact Results of T2 of C3 and M1 are C4 and M2
Impact Results of T3 of M2 and D1 are M3 and D2
Impact Result of T4 of D2 is D3
Impact Result of T5 of D2 is D4
Impact Result of T6 of D2 is D5
Impact Results of T7 of P2 and D3 are D6 and P3
Impact Results of T8 of P2 and D4 are D7 and P4
Impact Results of T9 of P2 and D5 are D8 and P5
Impact Results of T3 of M2 and D1 are M3 and D2

Рис. 5. Результат логического анализа концептуального графа логической сети Петри по внутреннему XML/PNML представлению

Начальная интерпретация предиката $R(a)$, $a \in A$, соответствует состояниям готовности участников судебного процесса к действиям:

$R(P_1)$ = **true**, $R(P_2)$ = **false**, $R(P_3)$ = **false**, $R(P_4)$ = **false**, $R(P_5)$ = **false**,

$R(C_1)$ = **true**, $R(C_2)$ = **false**, $R(C_3)$ = **false**, $R(C_4)$ = **false**, $R(M_1)$ = **true**,

$R(M_2)$ = **false**, $R(M_3)$ = **false** , $R(D_1)$ = **true**, $R(D_2)$ = **false**, $R(D_3)$ = **false**,

$R(D_4)$ = **false**, $R(D_5)$ = **false**, $R(D_6)$ = **false**, $R(D_7)$ = **false**, $R(D_8)$ = **false**.

Систему правил срабатывания переходов представим в следующем виде непосредственно по правилам на рис. 5:

$$URT_0 : R(P_1) \,\&\, R(C_1) \to R(P_2) \,\&\, R(C_2);$$

$$URT_1 : R(C_2) \to R(C_3);$$

$$URT_2 : R(C_3) \,\&\, R(M_1) \to R(C_4) \,\&\, R(M_2);$$

$$URT_3 : R(M_2) \,\&\, R(D_1) \to R(M_3) \,\&\, R(D_2).$$

Последующие правила для переходов T_4, T_5 и T_6 несовместимы; выбор одного из этих правил в логической модели формально может осуществить, например, «внешний» наблюдатель. В данном случае этот выбор зависит от поведения ответчика:

$$URT_4 : R(D_2) \to R(D_3);$$

$$URT_5 : R(D_2) \to R(D_4);$$

$$URT_6 : R(D_2) \to R(D_5).$$

В результате предыдущего выбора одного из правил срабатывания переходов T_4, T_5 или T_6 далее будет выполняться одно из соответствующих им правил URT_7, URT_8 или URT_9:

$$URT_7 : R(P_2) \,\&\, R(D_3) \to R(P_3) \,\&\, R(D_6);$$

$$URT_8 : R(P_2) \,\&\, R(D_4) \to R(P_4) \,\&\, R(D_7);$$

$$URT_9 : R(P_2) \,\&\, R(D_5) \to R(P_5) \,\&\, R(D_8).$$

В итоге получаем один из альтернативных ответов, о чем свидетельствует наличие меток в выходных позициях одного из сработавших переходов T_7, T_8 или T_9.

Рассмотрим, например, смысл выражения URT_0, которое соответствует срабатыванию перехода T_0:

$$URT_0 : R(P_1) \,\&\, R(C_1) \to R(P_2) \,\&\, R(C_2).$$

Это выражение означает, что если до срабатывания перехода T_0 было истинно выражение $R(P_1) \,\&\, R(C_1)$, то после его срабатывания будет истинно выражение $R(P_2) \,\&\, R(C_2)$.

Приведенные выше выражения полностью соответствуют концептуальному графу логической сети Петри. Однако эти данные дают лишь косвенное представление о перемещении меток, что не позволяет осуществить программную реализацию данной логической модели на процедурно ориентированных языках. Поэтому на основе данных выражений следует получить новые выражения для правил срабатывания переходов, в которых учитывается перемещение меток.

Следующие модификации предыдущих логических правил учитывают перемещение меток, естественно, выраженное в логической форме:

$$URT_0 : R(P_1) \,\&\, R(C_1) \to \neg R(P_1) \,\&\, \neg R(C_1) \,\&\, R(P_2) \,\&\, R(C_2);$$

$$URT_1 : R(C_2) \to \neg R(C_2) \,\&\, R(C_3);$$

$$URT_2 : R(C_3) \,\&\, R(M_1) \to \neg R(C_3) \,\&\, \neg R(M_1) \,\&\, R(C_4) \,\&\, R(M_2);$$

$$URT_3 : R(M_2) \,\&\, R(D_1) \to \neg R(M_2) \,\&\, \neg R(D_1) \,\&\, R(M_3) \,\&\, R(D_2).$$

Например, рассмотрим подробнее следующее правило:

$$URT_0 : R(P_1) \,\&\, R(C_1) \to \neg R(P_1) \,\&\, \neg R(C_1) \,\&\, R(P_2) \,\&\, R(C_2),$$

где истинности конъюнкции $R(P_1)\,\&R(C_1)$ соответствует наличие по одной метке в каждой из входных позиций перехода T_0. После срабатывания данного перехода эти метки покинут его входные позиции – это будет соответствовать тому, что оба элементарных высказывания $R(P_1)$ и $R(C_1)$ станут ложными, что и отражено в формуле $\neg R(P_1)\,\&\neg R(C_1)\,\&R(P_2)\,\&\,\&R(C_2)$. Истинность высказываний $R(P_2)$ и $R(C_2)$ соответствует появлению по одной метке в выходных полициях перехода T_0.

Как ранее было декларировано, последущие правила для переходов T_4, T_5 и T_6 несовместимы, т.е. истинным может быть только одно из них: условно выбор одного из этих правил в логической модели может осуществить, например, «внешний» наблюдатель (в реальности этот выбор, как и ранее, зависит от поведения ответчика):

$$URT_4 : R(D_2) \to \neg R(D_2)\,\&R(D_3);$$

$$URT_5 : R(D_2) \to \neg R(D_2)\,\&R(D_4);$$

$$URT_6 : R(D_2) \to \neg R(D_2)\,\&R(D_5).$$

В результате предыдущего выбора одного из правил срабатывания переходов T_4, T_5 или T_6 далее должно выполниться одно из соответствующих правил URT_7, URT_8 или URT_9:

$$URT_7 : R(P_2)\,\&R(D_3) \to \neg R(P_2)\,\&\neg R(D_3)\,\&R(P_3)\,\&R(D_6);$$

$$URT_8 : R(P_2)\,\&R(D_4) \to \neg R(P_2)\,\&\neg R(D_4)\,\&R(P_4)\,\&R(D_7);$$

$$URT_9 : R(P_2)\,\&R(D_5) \to \neg R(P_2)\,\&\neg R(D_5)\,\&R(P_5)\,\&R(D_8).$$

В итоге должен быть получен один из альтернативных несовместимых ответов, о чем свидетельствует наличие меток в выходных позициях одного из сработавших переходов T_7, T_8 или T_9, которым соответствуют следующие интерпретации:

T_7 – ОТВЕТЧИК официально отвечает ИСТЦУ, $R(P_3)$ = **true** – ИСТЕЦ получил официальный ответ от ОТВЕТЧИКА, $R(D_6)$ = **true** – ОТВЕТЧИК официально ответил ИСТЦУ;

T_8 – ОТВЕТЧИК выставляет встречный иск, $R(P_4)$ = **true** – ОТВЕТЧИК выставил встречный иск ИСТЦУ, $R(D_7)$ = **true** – ИСТЕЦ получил встречный иск от ОТВЕТЧИКА;

T_9 – ОТВЕТЧИК не является в суд, $R(P_5)$ = **true** – ИСТЕЦ не получил ответ от ОТВЕТЧИКА в заданный срок, $R(D_8)$ = **true** – ОТВЕТЧИК не явился в суд в назначенное время.

Продукционная модель представления знаний о прохождении искового заявления

Систему логических правил, носящих декларативный характер, можно далее реализовать на языке Prolog. Однако для программной реализации модели на каком-либо процедурно-ориентированном языке необходимо исполь-

зовать исполнимую модель представления знаний о предметной области. Под исполнимостью модели подразумевается тот факт, что она непосредственно пригодна для кодирования моделирующей программы. К подобным моделям относятся конечные автоматы, сети Петри. Продукционные модели представления знаний о предметной области содержат как декларативную составляющую, так и процедурную. Процедурную и «исполнимую» составляющие концептуальные логические сети Петри «заимствуют», естественно, от сетей Петри, лежащих в их основе.

Определим продукционную модель логической (бинарной) сети Петри следующим образом: элементы кортежа A, T и R определяются так же, как и для сети $CLPN_1$, а четвертым элементом кортежа является множество $PR = \{PRT_0, PRT_1, PRT_2, PRT_3, PRT_4, PRT_5, PRT_6, PRT_7, PRT_8, PRT_9\}$ продукционных правил (англ. *Production Rules*), модифицирующих предикат разметки позиций $R(a)$, $a \in A$, и определяющих порядок срабатывания переходов сети $CLPN_1$.

При построении системы продукционных правил воспользуемся рассмотренными ранее логическими правилами, которые описывали логику срабатывания переходов сети Петри. Использована следующая нотация для оператора, подобного оператору *if* в процедурно-ориентированных языках:

$$[\alpha](A \oplus B).$$

Данное выражение означает, что при истинности условия α выполняется действие A, а при его ложности – действие B. Здесь $[\alpha]$ – условная часть продукционного правила, где квадратные скобки обязательны. В круглые скобки, также обязательные, заключена операционная часть продукционного правила. Далее данное выражение будет использовано в следующем виде:

$$[\alpha](A \oplus E),$$

где E – «пустое» действие.

В правых частях продукционных правил реализуются действия по модификации предиката $R(a)$ при помощи операции присваивания ":=" (другое допустимое обозначение – обратная стрелка "←"). Выполнению продукционных правил должна предшествовать установка начальной разметки позиций, т.е. установка начальных значений предиката $R(a)$ сети $CLPN_1$:

$$R(P_1) := \textbf{true}, R(P_2) := \textbf{false}, R(P_3) := \textbf{false}, R(P_4) := \textbf{false}, R(P_5) := \textbf{false},$$

$$R(C_1) := \textbf{true}, R(C_2) := \textbf{false}, R(C_3) := \textbf{false}, R(C_4) := \textbf{false}, R(M_1) := \textbf{true},$$

$$R(M_2) := \textbf{false}, R(M_3) := \textbf{false}, R(D_1) := \textbf{true}, R(D_2) := \textbf{false}, R(D_3) := \textbf{false},$$

$$R(D_4) := \textbf{false}, R(D_5) := \textbf{false}, R(D_6) := \textbf{false}, R(D_7) := \textbf{false}, R(D_8) := \textbf{false}.$$

Далее задается последовательность продукционных правил:

$$PRT_0 : \left[R(P_1)\ \&\ R(C_1)\right]\left(R(P_1) := \neg R(P_1), R(C_1) := \neg R(C_1),\right.$$

$$\left. R(P_2) := \textbf{true}, R(C_2) := \textbf{true} \oplus E\right);$$

$$PRT_1 : \left[R(C_2)\right]\left(R(C_2) := \neg R(C_2), R(C_3) := \textbf{true} \oplus E\right);$$

$$PRT_2 :\left[R(C_3)\,\&R(M_1)\right]\left(R(C_3) := \neg R(C_3), R(M_1) := \neg R(M_1),\right.$$

$$\left. R(C_4) := \mathbf{true}, R(M_2) := \mathbf{true} \oplus E\right);$$

$$PRT_3 :\left[R(M_2)\,\&R(D_1)\right]\left(R(M_2) := \neg R(M_2), R(D_1) := \neg R(D_1),\right.$$

$$\left. R(M_3) := \mathbf{true}, R(D_2) := \mathbf{true} \oplus E\right).$$

Последующие продукционные правила для переходов T_4, T_5 и T_6 несовместимы, т.е. выполняться может только одно из них; выбор одного из этих правил в модели может осуществить пользователь, вводя значения переменной K:

$$PRT_4 :\left[(K=1)\,\&R(D_2)\right]\left(R(D_2) := \neg R(D_2), R(D_3) := \mathbf{true} \oplus E\right);$$

$$PRT_5 :\left[(K=2)\,\&R(D_2)\right]\left(R(D_2) := \neg R(D_2), R(D_4) := \mathbf{true} \oplus E\right);$$

$$PRT_6 :\left[(K=3)\,\&R(D_2)\right]\left(R(D_2) := \neg R(D_2), R(D_5) := \mathbf{true} \oplus E\right).$$

В результате предыдущего выбора одного из правил срабатывания переходов T_4, T_5 или T_6 далее будет выполняться одно из соответствующих им правил T_7, T_8 или T_9:

$$PRT_7 :\left[R(P_2)\,\&R(D_3)\right]\left(R(P_2) := \neg R(P_2), M(D_3) := \neg M(D_3),\right.$$

$$\left. R(P_3) := \mathbf{true}, R(D_6) \oplus \mathbf{true}\ E\right);$$

$$PRT_8 :\left[R(P_2)\,\&R(D_4)\right]\left(R(P_2) := \neg R(P_2)\ \&\ R(D_4),\right.$$

$$\left. R(P_4) := \mathbf{true},\ R(D_7) \oplus \mathbf{true}\ E\right);$$

$$PRT_9 :\left[R(P_2)\,\&R(D_5)\right]\left(R(P_2) := \neg R(P_2), R(D_5) := \neg R(D_5),\right.$$

$$\left. R(P_5) := \mathbf{true}, R(D_8) \oplus \mathbf{true}\ E\right).$$

Примечание. В операционной части продукционных правил операции модификации предиката R разделены запятыми, означающими последовательное выполнение операций.

В итоге исполнения модели формируется один из альтернативных ответов, о чем свидетельствует наличие меток в выходных позициях одного из сработавших переходов T_7, T_8 или T_9.

Возможна ситуация, когда по каким-то причинам не удается проверить исходную сеть Петри на безопасность (1-ограниченность). Тогда для гарантии того, что моделируемая сеть Петри является логической, достаточно замкнуть ингибиторные связи каждого перехода со своими выходными позициями, тем самым проверяя эти позиции на отсутствие меток перед срабатыванием перехода. Тогда система продукционных правил PR получит следующий вид:

$$PRT_0 :\left[R(P_1)\,\&R(C_1)\,\&\neg R(P_2)\,\&\,\neg R(C_2)\right](R(P_1) := \neg R(P_1),$$

$$R(C_1) := \neg R(C_1), R(P_2) := \mathbf{true}, R(C_2) := \mathbf{true} \oplus E) ;$$

$$PRT_1 :\left[R(C_2)\,\&\neg R(C_3)\right]\left(R(C_2) := \neg R(C_2), R(C_3) := \mathbf{true} \oplus E\right);$$

$$PRT_2 :\left[R(C_3)\,\&R(M_1)\,\&\neg R(C_4)\,\&\neg R(M_2)\right](R(C_3) := \neg R(C_3),$$

$$R(M_1) := \neg R(M_1),\ R(C_4) := \mathbf{true},\ R(M_2) := \mathbf{true} \oplus E);$$

$$PRT_3 :\left[R(M_2)\,\&R(D_1)\,\&\neg R(M_3)\,\&\neg R(D_2)\right](R(M_2) := \neg R(M_2),$$

$$R(D_1) := \neg R(D_1),\ R(M_3) := \mathbf{true},\ R(D_2) := \mathbf{true} \oplus E);$$

$$PRT_4 :\left[(K = 1)\,\&R(D_2)\,\&\neg R(D_3)\right]$$

$$\left(R(D_2) := \neg R(D_2), R(D_3) := \mathbf{true} \oplus E\right);$$

$$PRT_5 :\left[(K = 2)\,\&R(D_2)\,\&\neg R(D_4)\right]$$

$$\left(R(D_2) := \neg R(D_2), R(D_4) := \mathbf{true} \oplus E\right);$$

$$PRT_6 :\left[(K = 3)\,\&R(D_2)\,\&\neg R(D_5)\right]$$

$$\left(R(D_2) := \neg R(D_2), R(D_5) := \mathbf{true} \oplus E\right);$$

$$PRT_7 :\left[R(P_2)\,\&R(D_3)\,\&\neg R(P_3)\,\&\neg R(D_6)\right](R(P_2) := \neg R(P_2),$$

$$M(D_3) := \neg M(D_3),\ R(P_3) := \mathbf{true},\ R(D_6) := \mathbf{true} \oplus E);$$

$$PRT_8 :\left[R(P_2)\,\&R(D_4)\,\&\neg R(P_4)\,\&\neg R(D_7)\right](R(P_2) := \neg R(P_2)\,\&\neg R(D_4),$$

$$R(P_4) := \mathbf{true},\ R(D_7) := \mathbf{true} \oplus E);$$

$$PRT_9 :\left[R(P_2)\,\&R(D_5)\,\&\neg R(P_5)\,\&\neg R(D_8)\right](R(P_2) := \neg R(P_2),$$

$$R(D_5) := \neg R(D_5),\ R(P_5) := \mathbf{true}, R(D_8) := \mathbf{true} \oplus E).$$

В соответствии с данной системой правил составлена программа «Иск», текст которой приведен в конце настоящей работы.

Развитие концептуальной модели правового процесса

В работах [20, 29] предполагалось, что «моделирование юридических систем сетями Петри могло бы привести к лучшему пониманию их, а анализ модели мог бы привести к улучшению юридической системы». Следующий пример показывает, что представление модели правового процесса концептуальной логической сетью Петри позволяет более полно учитывать особенности предметной области за счет усиления декларативной составляющей мо-

дели представления знаний и, как следствие, повысить результативность использования модели.

Концептуальный граф логической сети Петри может включать элементы, наличествующие в традиционных концептуальных графах. Например, в графе КЛ СП на рис. 6 доопределены функции участников правового процесса. Результат логического анализа фрагмента концептуального графа логической сети Петри на рис. 6 по внутреннему XML/PNML представлению представлен на рис. 7.

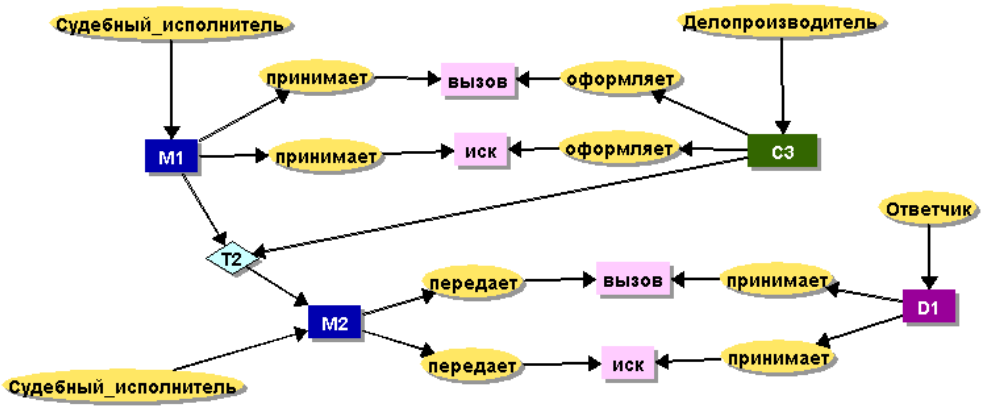


Рис. 6. Дополненный фрагмент концептуального графа логической сети Петри (КЛ СП)

There are the Propositions where:
Делопроизводитель is C3
оформляет of C3 is иск
оформляет of C3 is вызов
Судебный_исполнитель is M1
принимает of M1 is вызов
принимает of M1 is иск
Судебный_исполнитель is M2
передает of M2 is иск
передает of M2 is вызов
Ответчик is D1
принимает of D1 is иск
принимает of D1 is вызов
Impact Result T2 of M1 and C3 is M2

Рис. 7. Результат логического анализа фрагмента концептуального графа логической сети Петри по внутреннему XML/PNML представлению

Дополненному фрагменту концептуального графа логической сети Петри соответствуют получаемые автоматически следующие высказывания (факты) и правило срабатывания перехода T_2 (рис. 8).

Полученные структурированные знания могут быть использованы при дальнейшей реализации интеллектуальной экспертной системы, сопровождающей правовой процесс.

There are the Propositions where:

Делопроизводитель(C3),
оформляет(C3, иск),
оформляет(C3, вызов).
Судебный_исполнитель(M1),
принимает(M1, вызов),
принимает(M, иск).
Судебный_исполнитель(M2),
передает(M2, иск),
передает(M2, вызов).
Ответчик(D1),
Принимает(D1, иск),
Принимает(D1, вызов).
Правило срабатывания перехода:
T2: M1 & C3 → M2.

Рис. 8. Факты и правило, формируемые автоматически по концептуальному графу на рис. 6

Программа «Иск» на языке Python, реализующая концептуальную логическую сеть Петри $CLPN_1$ для прототипа части интеллектуальной системы, сопровождающей правовой процесс

Продукционные правила пригодны для их использования в качестве исполнимых спецификаций для компьютерной программы; другими словами, они легко программируются. Приведенная на рис. 9 программа на языке Python составлена с учетом того факта, что выражения с предикатным символом $R(a)$, $a \in A$ в системе продукционных правил представляют собой атомарные высказывания. Поэтому в программе вместо них используются булевы переменные, одноименные позициям концептуальной логической сети Петри.

Данный пример имеет демонстрационный характер и результаты его исследования и реализации могут после некоторых модификаций использоваться и для других областей человеческой деятельности, где участники ведут диалог, участвуют в работе над общим проектом, высказывают свои мнения о дальнейшем продвижении проекта.

Пример построения предметно интерпретируемой концептуальной логической сети Петри, моделирующей строительство промышленного объекта

В работе [20] приведен пример сети Петри для строительства дома; в качестве контрпримера была выбрана PERT-диаграмма (англ. *Program* (*Project*) *Evaluation and Review Technique*) – метод оценки и анализа проектов, который используется в управлении проектами [30]. Каждый этап PERT-диаграммы представлен позицией, а причинно-следственные связи – переходами. В отличие от сети Петри, приведенной в работе [20], описание процесса строительства дома расширено за счет указания названий этапов. На рис. 10 представлен концептуальный граф логической сети Петри, опи-

сывающей процесс строительства дома. Индексы при именах объектов показывают время реализации этапа.

Программа «Иск» (начало)

```
p1=d1=m1=c1=True
p2=p3=p4=p5=c2=c3=c4=d2=False
d3=d4=d5=d6=d7=d8=m2=m3=False
choice=0
if(p1 and c1):
   p1=not(p1)
   c1=not(c1)
   p2=True
   c2=True
   print("t0")
if(c2):
   c2=not(c2)
   c3=True
   print("t1")
 if(c3 and m1):
   c3=not(c3)
   m1=not(m1)
   c4=True
   m2=True
   print("t2")
if(m2 and d1):
   m2=not(m2)
   d1=not(d1)
   m3=True
   d2=True
   print("t3")
print("Введите номер ветки, по
которой будет выполняться
дальнейший переход: 1\t 2\t 3\t")
choice=input()
      print("t7")
   print("Ответчик официально
отвечает истцу")
```

Программа «Иск» (окончание)

```
if(choice=="1"):
   d2=not(d2)
   d3=True
   print("t4")
   if(p2 and d3):
      p2=not(p2)
      d3=not(d3)
      p3=True
      d6=True
elif(choice=="2"):
   d2=not(d2)
   d4=True
   print("t5")
   if(p2 and d4):
      p2=not(p2)
      d4=not(d4)
      p4=True
      d7=True
      print("t8")
   print("Ответчик выставляет
встречный иск")
elif(choice=="3"):
   d2=not(d2)
   d5=True
   print("t6")
   if(p2 and d5):
      p2=not(p2)
      d5=not(d5)
      p5=True
      d8=True
      print("t9")
   print("Ответчик не является
в суд")
else:
   print("Вы ввели неверные
данные!")
```

Рис. 9. Программа моделирования концептуальной логической сети Петри

В случае использования временных сетей Петри могут быть естественно указаны моменты начала и окончания выполнения каждого этапа, что позволит получить те же временные характеристики, которые определяются по PERT-диаграмме. Могут быть указаны также материалы и число рабочих, занятых выполнением каждого этапа.

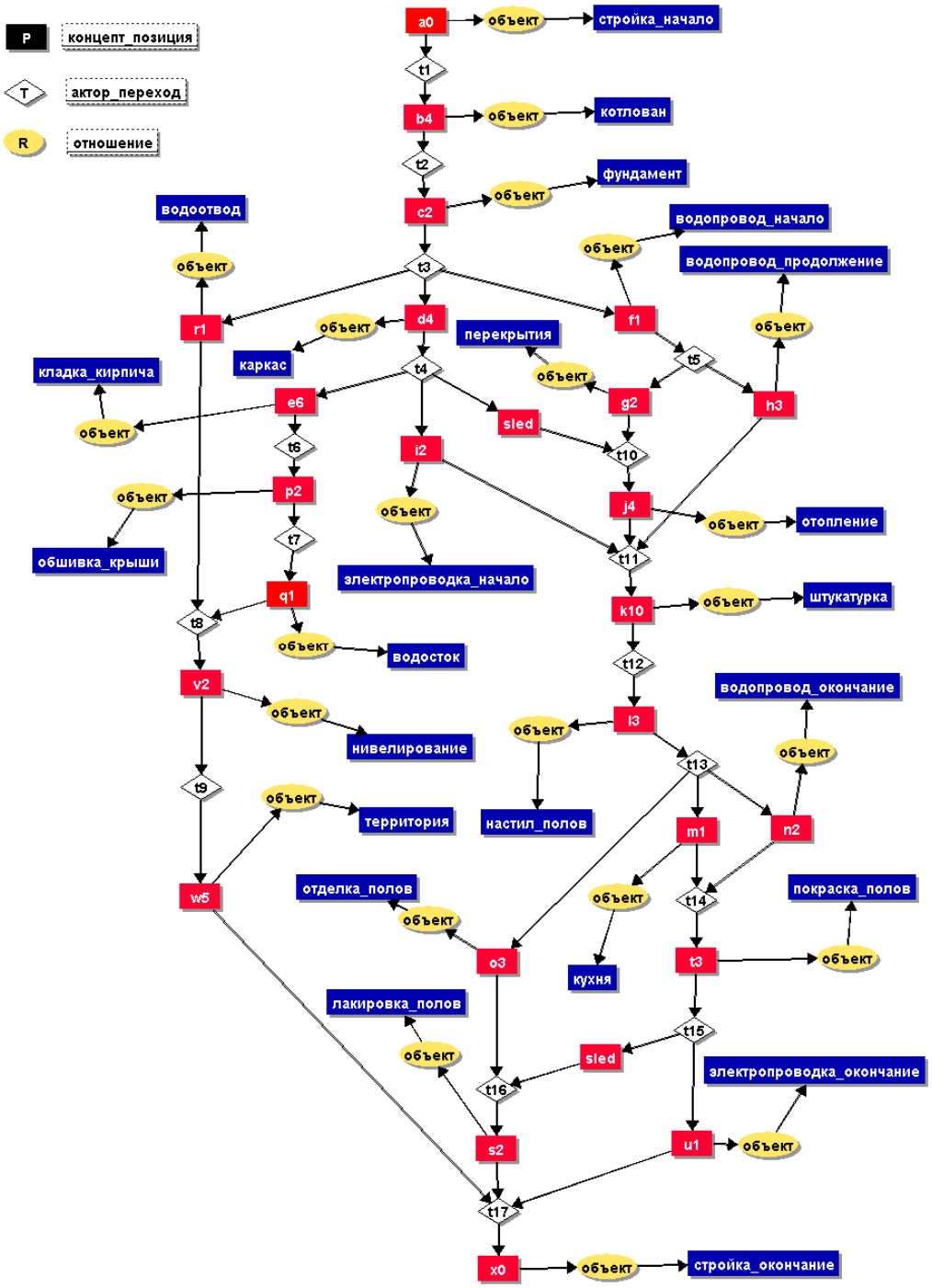


Рис. 10. Концептуальный граф логической сети Петри, описывающей процесс строительства

Факты и правила получены автоматически путем трансформации КЛ СП на рис. 11 в текст, содержащий факты и правила, т.е. цель исследования выполнена.

Facts	Rules
объект of a0 is стройка_начало	t1 of a0 is b4
объект of b4 is котлован	t2 of b4 is c2
объект of c2 is фундамент	t3 of c2 are f1 and r1 and d4
объект of d4 is каркас	t4 of d4 are sled and e6 and i2
объект of r1 is водоотвод	t5 of f1 are h3 and g2
объект of f1 is водопровод_начало	t6 of e6 is p2
объект of e6 is кладка_кирпича	t7 of p2 is q1
объект of i2 is электропроводка_начало	t8 of q1 and r1 is v2
объект of g2 is перекрытия	t9 of v2 is w5
объект of h3 is водопровод_продолжение	t10 of g2 and sled is j4
объект of p2 is обшивка_крыши	t11 of i2 and j4 and h3 is k10
объект of q1 is водосток	t12 of k10 is l3
объект of j4 is отопление	t13 of l3 are o3 and n2 and m1
объект of k10 is штукатурка	t14 of m1 and n2 is t3
объект of v2 is нивелирование	t15 of t3 are sled and u1
объект of l3 is настил_полов	t16 of sled and o3 is s2
объект of w5 is территория	t17 of s2 and w5 and u1 is x0
объект of m1 is кухня	
объект of n2 is водопровод_окончание	
объект of o3 is отделка_полов	
объект of t3 is покраска_полов	
объект of s2 is лакировка_полов	
объект of u1 is электропроводка_окончание	
объект of x0 is стройка_окончание	

Рис. 11. Факты и правила выполнения автоматической трансформации

Заключение

Формально определены концептуальные логические сети Петри. Показано, что они могут использоваться при построении декларативных, логических и исполнимых продукционных моделей представления знаний. Логические и продукционные модели легко программируются, а реализованные программы могут быть прототипами для построения на их основе различного рода справочных и экспертных систем, применяемых в различных отраслях науки и техники. Реализована и проиллюстрирована примерами методика синтеза концептуальных логических сетей Петри на основе выявления общей семантики концептуальных графов и сетей Петри, в результате чего построены модели, обладающие декларативными, императивными и динамическими свойствами. Автоматизировано выделение фактов и правил вывода при последующей программной реализации данного подхода в интеллектуальных событийных системах на примере конкретных предметных областей человеческой деятельности. Подобно Semantic Web, основой для представления интеллектуальной информации о предметной области являются семантическая сеть или концептуальный граф с событиями, пригодные для последующей машинной обработки.

Список литературы

1. История информатики и философия информационной реальности : учеб. пособие для вузов / под ред. чл.-корр. РАН Р. М. Юсупова, проф. В. П. Котенко. М. : Академический проект, 2007. 429 с.

2. Berners-Lee T., Fensel D., Hendler J. A. [et al.]. Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential. The MIT Press, 2005.
3. Uniform Resource Identifier. [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/URI (дата обращения: 31.01.2023).
4. Хорошевский В. Ф. Проектирование систем программного обеспечения под управлением онтологий: модели, методы, реализации // Онтология проектирования. 2019. Т. 9, № 4 (34). С. 429–448. doi: 10.18287/2223-9537-2019-9-4-429-448
5. Project MOST Marrying Ontology and Software Technology. 2011. URL: https://cordis.europa.eu/project/rcn/85351/factsheet/en (дата обращения: 31.01.2023).
6. Волчихин В. И., Карамышева Н. С., Митрохин М. А., Зинкин С. А. Представление и структурирование знаний в семантико-ориентированной вычислительной среде. Часть I. Интеграция концептуальных графов и логических сетей на основе формализации структурированных ситуаций // Известия высших учебных заведений. Поволжский регион. Технические науки. 2023. № 2. С. 24–51.
7. Поспелов Д. А. Ситуационное управление: теория и практика. М. : Наука, 1986. 288 с.
8. Брой М. Информатика. Основополагающее введение. М. : Диалог-МИФИ, 1996. Ч. 1. 299 с.
9. Аврамчук Е. Ф., Вавилов А. А., Емельянов С. В. [и др.]. Технология системного моделирования. М. : Машиностроение ; Берлин : Техник, 1988. 520 с.
10. Зинкин С. А. Элементы технологии иерархического концептуального моделирования и реализации систем и сетей хранения и обработки данных // Известия высших учебных заведений. Поволжский регион. Технические науки. 2008. № 4. С. 3–15.
11. Зинкин С. А. Иерархические сети абстрактных машин и виртуализация интеллектуальных систем внешнего хранения и обработки данных // Известия высших учебных заведений. Поволжский регион. Технические науки. 2009. № 2. С. 25–38.
12. Волчихин В. И., Зинкин С. А. Логико-алгебраические модели и методы в проектировании функциональной архитектуры распределенных систем хранения и обработки данных // Известия высших учебных заведений. Поволжский регион. Технические науки. 2012. № 2. С. 3–16.
13. Ершов Ю. Л., Палютин Е. А. Математическая логика. М. : Физматлит, 2011. 357 с.
14. Плоткин Б. И. Универсальная алгебра, алгебраическая логика и базы данных. М. : Наука, 1991. 448 с.
15. Тейз А., Грибомон П., Луи Ж. [и др.]. Логический подход к искусственному интеллекту: от классической логики к логическому программированию. М. : Мир, 1990. 429 с.
16. Polovina S. An Introduction to Conceptual Graphs. Conceptual Structures: Knowledge Architectures for Smart Applications / ed. by U. Priss, S. Polovina, R. Hill // Lecture Notes in Artificial Intelligence (LNAI 4604). Springer, 2007. P. 1–15.
17. Палагин А. В., Кривой С. Л., Петренко Н. Г. Концептуальные графы и семантические сети в системах обработки естественно-языковой информации // Математические машины и системы. 2009. № 3. С. 67–79.
18. A World of Conceptual Graphs. [Электронный ресурс]. URL: http://conceptualgraphs.org/ (дата обращения: 31.01.2023).
19. Глушков В. М., Цейтлин Г. Е., Ющенко Е. Л. Методы символьной мультиобработки. Киев : Наукова думка, 1980. 252 с.
20. Питерсон Дж. Теория сетей Петри и моделирование систем. М. : Мир, 1984. 254 с.
21. Котов В. Е. Сети Петри. М. : Наука, 1984. 160 с.
22. Мараховский В. Б., Розенблюм Л. Я., Яковлев А. В. Моделирование параллельных процессов. Сети Петри. СПб. : Профессиональная литература, 2014. 400 с.
23. CharGer Manual v3.5b1 2005-11-30. P. 1–58. [Электронный ресурс]. URL: http://charger.sourceforge.net/ (дата обращения: 31.01.2023).

24. Delugah H. CharGer – A Conceptual Graph Editor written by Harry Delugah [Электронный ресурс]. URL: http://www.cs.uah.edu/~delugach/CharGer/ (дата обращения: 31.01.2023).
25. Зинкин С. А., Пащенко Д. В., Пучкова У. Н., Джафар М. С. Интеграция методов концептуального и поведенческого моделирования дискретно-событийных систем: II. Логико-алгебраические операционные модели и инфокоммуникационные технологии // Кибернетика и программирование. 2017. № 1. С. 75–93.
26. Мустафа С. Д., Зинкин С. А., Карамышева Н. С. Организация функционирования распределенных вычислительных систем с переменной архитектурой в виде облачного сервиса, формируемого по запросу клиента (концептуальные графы распределенных алгоритмов) // XXI век: итоги прошлого и проблемы настоящего плюс. 2018. Т. 7, № 4 (44). С. 136–146.
27. Зинкин С. А., Мустафа С. Д., Карамышева Н. С. Концептуальные представления и модификации сетей Петри для приложений в области синтеза функциональной архитектуры распределенных вычислительных систем с переменной структурой // Известия Юго-Западного государственного университета. 2018. Т. 22, № 6 (81). С. 143–167.
28. Graphical Evaluation and Review Technique Simulation. [Электронный ресурс]. URL: https://cyberpedia.su/9x12977.html (дата обращения: 31.01.2023).
29. Meldman J. A Petri-Net Representation Civil Procedure // The Journal of Law and Technology. 1978. Vol. 19, № 2. P. 123–148.
30. Program (Project) Evaluation and Review Technique (PERT). [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/PERT (дата обращения: 31.01.2023).

References

1. Yusupova R.M., Kotenko V.P. (eds.). *Istoriya informatiki i filosofiya informatsionnoy real'nosti: ucheb. posobie dlya vuzov = History of computer science and philosophy of information reality: textbook*. Moscow: Aka-demicheskiy proekt, 2007:429. (In Russ.)
2. Berners-Lee T., Fensel D., Hendler J.A. et al. *Spinning the Semantic Web: Bringing the World Wide Web to Its Full Potential*. The MIT Press, 2005.
3. *Uniform Resource Identifier*. Available at: https://ru.wikipedia.org/wiki/URI (accessed 31.01.2023).
4. Khoroshevskiy V.F. Design of software systems under ontology control: models, methods, implementations. *Ontologiya proektirovaniya = Design ontology*. 2019;9(4):429–448. (In Russ.). doi: 10.18287/2223-9537-2019-9-4-429-448
5. *Project MOST Marrying Ontology and Software Technology*. 2011. Available at: https://cordis.europa.eu/project/rcn/85351/factsheet/en (accessed 31.01.2023).
6. Volchikhin V.I., Karamysheva N.S., Mitrokhin M.A., Zinkin S.A. Representation and structuring of knowledge in the semantic oriented computing environment. Part 1. Intergration conceptual graphs and logical networks based on formalization of structured situations. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2023;(2):24–51. (In Russ.)
7. Pospelov D.A. *Situatsionnoe upravlenie: teoriya i praktika = Situational management: theory and practice*. Moscow: Nauka, 1986:288. (In Russ.)
8. Broy M. *Informatika. Osnovopolagayushchee vvedenie = Computer science. Basic Introduction*. Moscow: Dialog-MIFI, 1996;1:299. (In Russ.)
9. Avramchuk E.F., Vavilov A.A., Emel'yanov S.V. et al. *Tekhnologiya sistemnogo modelirovaniya = System Modeling Technology.* Moscow: Mashinostroenie; Berlin: Tekhnik, 1988:520. (In Russ.)
10. Zinkin S.A. Elements of technology for hierarchical conceptual modeling and implementation of data storage and processing systems and networks. *Izvestiya vysshikh*

uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences. 2008;(4):3–15. (In Russ.)
11. Zinkin S.A. Hierarchical networks of abstract machines and virtualization of intelligent systems for external data storage and processing. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2009;(2):25–38. (In Russ.)
12. Volchikhin V.I., Zinkin S.A. Logic-algebraic models and methods in the design of functional architecture of distributed data storage and processing systems. *Izvestiya vysshikh uchebnykh zavedeniy. Povolzhskiy region. Tekhnicheskie nauki = University proceedings. Volga region. Engineering sciences*. 2012;(2):3–16. (In Russ.)
13. Ershov Yu.L., Palyutin E.A. *Matematicheskaya logika = Mathematical logic.* Moscow: Fizmatlit, 2011:357. (In Russ.)
14. Plotkin B.I. *Universal'naya algebra, algebraicheskaya logika i bazy dannykh = Universal algebra, algebraic logic and databases*. Moscow: Nauka, 1991:448. (In Russ.)
15. Teyz A., Gribomon P., Lui Zh. et al. *Logicheskiy podkhod k iskusstvennomu intellektu: ot klassicheskoy logiki k logicheskomu programmirovaniyu = Logical approach to artificial intelligence: from classical logic to logic programming*. Moscow: Mir, 1990:429. (In Russ.)
16. Polovina S. An Introduction to Conceptual Graphs. Conceptual Structures: Knowledge Architectures for Smart Applications / ed. by U. Priss, S. Polovina, R. Hill. *Lecture Notes in Artificial Intelligence (LNAI 4604)*. Springer, 2007:1–15.
17. Palagin A.V., Krivoy S.L., Petrenko N.G. Conceptual graphs and semantic networks in natural language information processing systems. *Matematicheskie mashiny i sistemy = Mathematical machines and systems*. 2009;(3):67–79. (In Russ.)
18. *A World of Conceptual Graphs*. Available at: http://conceptualgraphs.org/ (accessed 31.01.2023).
19. Glushkov V.M., Tseytlin G.E., Yushchenko E.L. *Metody simvol'noy mul'tiobrabotki = Symbolic multiprocessing methods*. Kiev: Naukova dumka, 1980:252. (In Russ.)
20. Piterson J. *Teoriya setey Petri i modelirovanie system = Petri net theory and system modeling*. Moscow: Mir, 1984:254. (In Russ.)
21. Kotov V.E. *Seti Petri = Petri nets*. Moscow: Nauka, 1984:160. (In Russ.)
22. Marakhovskiy V.B., Rozenblyum L.Ya., Yakovlev A.V. *Modelirovanie parallel'nykh protsessov. Seti Petri = Petri nets*. Saint Petersburg: Professional'naya literatura, 2014:400. (In Russ.)
23. *CharGer Manual v3.5b1 2005-11-30*. P. 1–58. Available at: http://charger.sourceforge.net/ (accessed 31.01.2023).
24. Delugah H. *CharGer – A Conceptual Graph Editor written by Harry Delugah*. Available at: http://www.cs.uah.edu/~delugach/CharGer/ (accessed 31.01.2023).
25. Zinkin S.A., Pashchenko D.V., Puchkova U.N., Dzhafar M.S. Integration of methods for conceptual and behavioral modeling of discrete event systems: II. Logical-algebraic operating models and infocommunication technologies. *Kibernetika i programmirovanie = Cybernetics and programming*. 2017;(1):75–93. (In Russ.)
26. Mustafa S.D., Zinkin S.A., Karamysheva N.S. Organization of the functioning of distributed computing systems with variable architecture in the form of a cloud service formed at the request of the client (conceptual graphs of distributed algorithms). *XXI vek: itogi proshlogo i problemy nastoyashchego plyus = The 21st century: results of the past and problems of the present plus*. 2018;7(4):136–146. (In Russ.)
27. Zinkin S.A., Mustafa S.D., Karamysheva N.S. Conceptual representations and modifications of Petri nets for applications in the field of synthesis of functional architecture of distributed computing systems with variable structure. *Izvestiya Yugo-Zapadnogo gosudarstvennogo universiteta = Proceedings of the Southwest State University.* 2018;22(6):143–167. (In Russ.)

28. *Graphical Evaluation and Review Technique Simulation*. Available at: https://cyberpedia.su/9x12977.html (accessed 31.01.2023).
29. Meldman J. A Petri-Net Representation Civil Procedure. *The Journal of Law and Technology*. 1978;19(2):123–148.
30. *Program (Project) Evaluation and Review Technique (PERT)*. Available at: https://ru.wikipedia.org/wiki/PERT (accessed 31.01.2023).

Информация об авторах / Information about the authors

Владимир Иванович Волчихин
доктор технических наук, профессор, президент Пензенского государственного университета (Россия, г. Пенза, ул. Красная, 40)

E-mail: cnit@pnzgu.ru

Vladimir I. Volchikhin
Doctor of engineering sciences, professor, president of Penza State University (40 Krasnaya street, Penza, Russia)

Надежда Сергеевна Карамышева
кандидат технических наук, доцент, доцент кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: karamyshevans@yandex.ru

Nadezhda S. Karamysheva
Candidate of engineering sciences, associate professor, associate professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Максим Александрович Митрохин
доктор технических наук, доцент, заведующий кафедрой вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: vt@pnzgu.ru

Maksim A. Mitrokhin
Doctor of engineering sciences, associate professor, head of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Сергей Александрович Зинкин
доктор технических наук, профессор, профессор кафедры вычислительной техники, Пензенский государственный университет (Россия, г. Пенза, ул. Красная, 40)

E-mail: zsa49@yandex.ru

Sergey A. Zinkin
Doctor of engineering sciences, professor, professor of the sub-department of computer engineering, Penza State University (40 Krasnaya street, Penza, Russia)

Авторы заявляют об отсутствии конфликта интересов / The authors declare no conflicts of interests.

Поступила в редакцию / Received 21.05.2023

Поступила после рецензирования и доработки / Revised 19.07.2023

Принята к публикации / Accepted 20.08.2023