ВЫЧИСЛИТЕЛЬНАЯ ТЕХНИКА И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.031.6

Евгений Рафаилович Пантелеев
ФГБОУ ВО «Ивановский государственный энергетический университет имени В.И. Ленина», доктор технических наук, профессор кафедры программного обеспечения компьютерных систем, Россия, Иваново,
телефон (4932) 26-98-60, e-mail: erp@poks.ispu.ru

Арман Арсенович Мукучян
ФГБОУ ВО «Ивановский государственный энергетический университет имени В.И. Ленина», аспирант кафедры программного обеспечения компьютерных систем, Россия, Иваново, e-mail: 15508@gapps.ispu.ru

Логическая модель построения пошаговой контекстной помощи пользователю САПР

Авторское резюме

Состояние вопроса. Автоматическое формирование пошаговой контекстной помощи пользователю САПР существенно уменьшает время, необходимое для решения прикладной задачи, за счет исключения затрат на поиск необходимых подсказок в сопроводительной документации. Сети Петри являются адекватной моделью представления возможных действий пользователя с учетом состояния данных приложения (контекстом). Применение метода инверсии сети Петри, использующего для построения цепочек рекомендованных действий *ограниченный* перебор, предпочтительнее использования стандартной процедуры анализа достижимости, основанной на *исчерпывающем* переборе. Однако отсутствие в известных реализациях явного отделения аксиом инверсии (знаний) от механизма их обработки (вывода) лишает систему контекстной помощи необходимой гибкости при изменении аксиоматики для учета допущений, связанных с конкретной моделью. В связи с этим целью исследования является обеспечение необходимой гибкости системы контекстной помощи за счет отделения модели представления знаний от механизма вывода.
Материалы и методы. В качестве модели сценариев действий пользователя используется раскрашенная сеть Петри. Аксиомы инверсии реализованы на языке PROLOG. В качестве инструмента построения цепочек рекомендованных действий использован стандартный механизм вывода языка PROLOG.
Результаты. Предложена аксиоматическая модель инверсии сети Петри и реализованный на ее базе метод построения пошаговой контекстной помощи стандартной машиной вывода языка PROLOG. Метод отличается явным отделением знаний (аксиом инверсии) от механизма вывода (построения пошаговых рекомендаций), что уменьшает затраты на адаптацию системы контекстной помощи при внесении изменений в аксиоматику инверсии.
Выводы. Предложенный метод позволяет снизить временные затраты на адаптацию системы контекстной помощи, так как сфера вносимых изменений ограничена декларациями аксиом инверсии. Достоверность результатов подтверждается данными использования предложенного метода для формирования контекстной помощи пользователю САПР «Модель и Архив» отечественной компании CSoft. Полученные результаты позволят создавать контекстные справочные службы для существующих приложений с минимальными изменениями их кодовой базы.

Ключевые слова: пошаговая контекстная помощь, сети Петри, язык логического программирования PROLOG, аксиомы инверсии

Вестник ИГЭУ, 2023, вып. 3, с. 68–78.

Evgeny Rafailovich Panteleev
Ivanovo State Power Engineering University, Doctor of Engineering Sciences (Post-doctoral degree), Professor of Computer Systems Software Department, Russia, Ivanovo, telephone (4932) 26-98-60, e-mail: erp@poks.ispu.ru

Arman Arsenovich Mukuchyan
Ivanovo State Power Engineering University, Postgraduate Student of Computer Systems Software Department, Russia, Ivanovo, e-mail: 15508@gapps.ispu.ru

Logical model of stepwise contextual help for CAD user

Abstract

Background. Automatic stepwise contextual help for CAD systems users reduces time to solve the application task since it saves time to search the prompt message in system documentation. Petri nets (PN) can be used to bind available actions of the user considering the state of application data (context). Application of the Petri net inversion method, which uses *limited* enumeration to construct chains of recommended actions is preferable than using the standard reachability analysis procedure based on *exhaustive* enumeration. However, the absence in the known implementations of an explicit separation of the axioms of inversion (knowledge) from the mechanism of their processing (inference) deprives the stepwise contextual help system of the necessary flexibility when changing the axioms to consider the assumptions associated with a particular model. Thus, the aim of this research is to provide the necessary flexibility of the contextual help system by separating the knowledge representation model from the inference engine.
Materials and methods. A colored PN is used as a model of user action scenarios. The inversion axioms are implemented in the PROLOG language. The standard inference engine of the PROLOG language is used as a tool to construct chains of recommended actions.
Results. The authors have proposed an axiomatic model of PN inversion and a method to construct a stepwise contextual help by the standard inference engine of the PROLOG language. The method differs by explicit separation of knowledge (inversion axioms) from the inference engine (stepwise recommendations). It reduces the computational costs of adapting the contextual help system when changing the inversion axioms.
Conclusion. The proposed method allows to reduce the time spent on adapting the contextual help system, since the field of the changes is limited by the declarations of the inversion axioms. The reliability of the results is confirmed since the proposed method of contextual help is used for the CAD “Model and Archive” user of CSoft company. The results obtained allow creating contextual help services for existing applications with minimal changes to their code base.

Key words: stepwise contextual help, Petri nets, logic programming language PROLOG, axioms of inversion

DOI: 10.17588/2072-2672.2023.3.068-078

Введение. Решение прикладной задачи в САПР зачастую требует от пользователя выстраивания цепочки необходимых действий методом проб и ошибок. Например, одной из задач информационной системы для поддержки жизненного цикла объектов капитального строительства и технологического оборудования промышленных предприятий CADLib Модель и Архив (МиА)[1] [1] отечественного разработчика ГК CSoft является проверка цифровой трехмерной модели, включающая, в частности, измерение расстояний между точками и объектами, создание заметок. Для измерения расстояния между двумя точками необходимо выбрать соответствующий пункт выпадающего меню «Инструменты». Для опытного пользователя данные действия очевидны, однако новый сотрудник не может связать известные ему действия с элементами интерфейса (рис. 1,а) и вынужден обращаться к справочным файлам либо просматривать все элементы строки меню в поисках нужного действия.

Известные системы контекстной помощи (СКП) в различных прикладных областях предполагают наличие моделей сценариев действий пользователя и текущего контекста приложения. В качестве примера можно привести интеллектуальную обучающую систему SlideTutor для диагностики заболеваний по визуальным признакам [2] с контекстной помощью в виде пошаговых инструкций и диагностики ошибочных шагов студента. Ограничение «глубины» контекстной помощи действиями, непосредственно доступными из текущего контекста, существенно снижает ценность данного подхода для его применения в САПР, так как проектировщик обычно планирует решение задачи в терминах проектных операций[2], которые реализуются цепочками действий с элементами интерфейса

[1] САПР МиА позволяет управлять процессом проектирования, осуществлять проверку 3D-моделей, информационную поддержку в процессе строительства и эксплуатации зданий, сооружений и оборудования, организовывать коллективный доступ и управление инженерными данными информационной модели, проверять наличие коллизий информационных моделей и т.д.; Руководство пользователя CADLib Модель и Архив. – CSoft Development, 2020. – 312 с.

[2] Согласно ГОСТ 22487-77, под проектной операцией понимается действие или формализованная совокупность действий, составляющих часть проектной процедуры, алгоритм которых остается неизменным для ряда проектных процедур, инвариантных к инструменту их реализации.

САПР. Кроме того, использование различных моделей для представления сценариев действий пользователя и текущего контекста (правил продукции и онтологий соответственно) усложняет вывод контекстных рекомендаций.

Этот последний недостаток преодолен в СКП, разработанной нами для уже упоминавшейся САПР МиА. Здесь в качестве модели представления сценариев действий из текущего контекста были использованы сети Петри (СП) [3], применение которых полностью оправдало себя в системах управления потоками работ – workflow [4].

СКП МиА отображает в иерархическом списке доступные из текущего контекста действия с возможностью быстрого перехода к соответствующему пункту справки (рис. 1,б,в). Представленный на рис. 1 фрагмент иллюстрирует ранее описанную операцию измерения расстояния между точками. Предполагается, что пользователь, не знающий, как выполнить то или иное действие, вызывает СКП (стрелка №1). В отображаемом списке уровень абстракции операций уменьшается при движении вниз по иерархии. На нижнем уровне представлены элементарные действия (нажатие кнопок и т.п.). После того как пользователь нашел интересующую его операцию, он может открыть справочный файл с необходимым позиционированием с помощью СКП (стрелка №2) и выполнить описанное действие (стрелка №3).

Список действий обновляется при изменении состояния графического интерфейса либо данных проекта. Например, после выбора пункта меню «Расстояние между точками» пользователю предоставляется возможность выбрать первую точку (рис. 2,а) и СКП обновляет список доступных действий (рис. 2,б). Можно сделать вывод, что отображаются все прикладные задачи, доступные из текущего контекста с указанием первого необходимого элементарного действия, что сокращает время доступа к необходимым пунктам справочного файла.

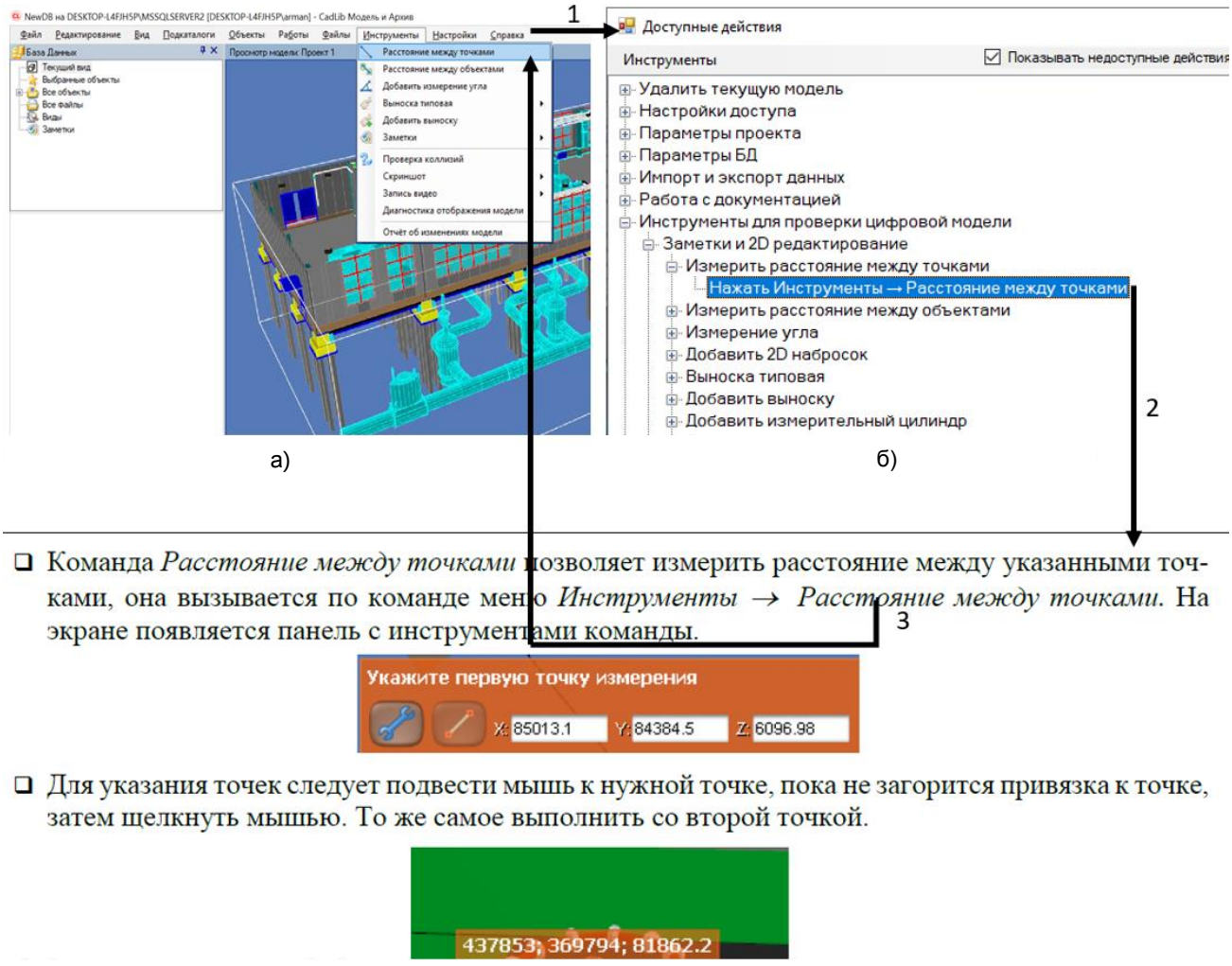
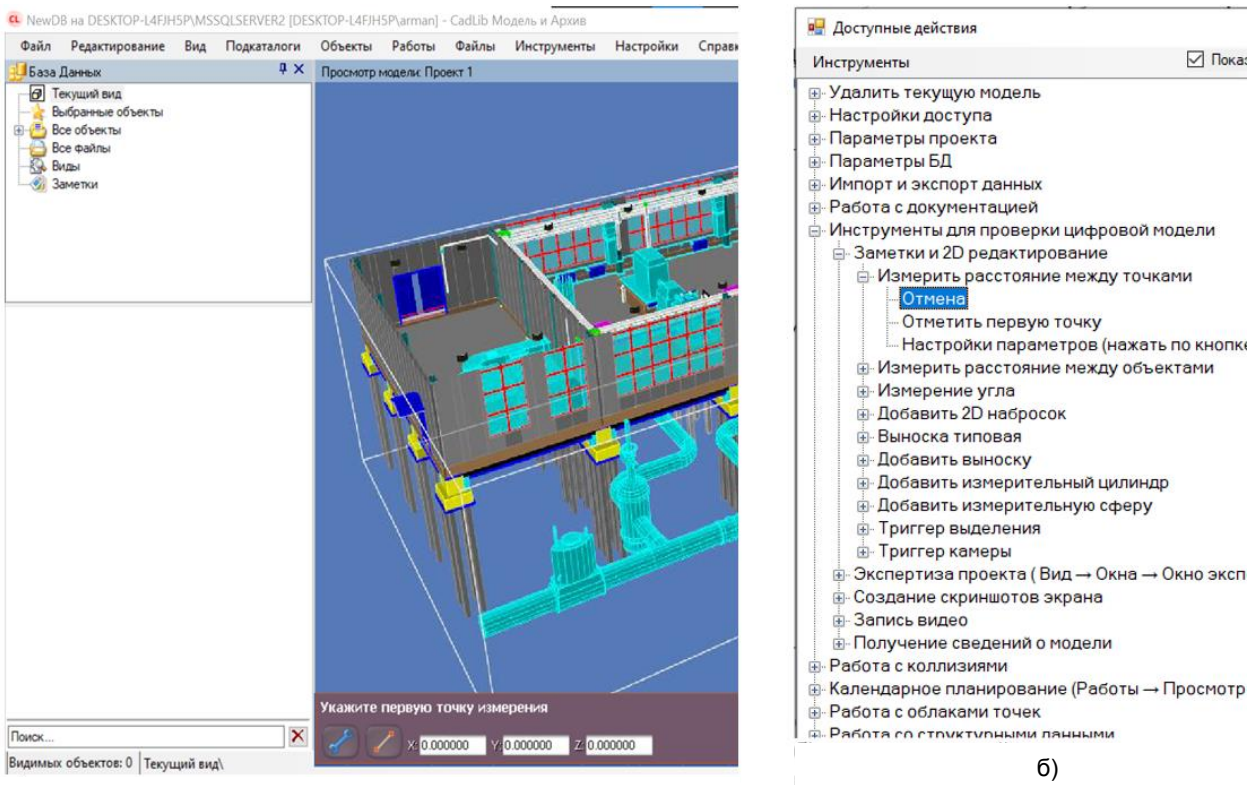


Рис. 1. Измерение расстояния между точками: а – интерфейс МиА. Просмотр модели; б – доступные действия для текущего контекста; в – фрагмент справочного файла для действия «Расстояние между точками»



а) б)

Рис. 2. Измерение расстояния между точками: а – измерение расстояния между точками; б – доступные действия для текущего контекста

В рамках обсуждавшихся подходов СКП отображают текущий контекст приложения на позиции справочного ресурса, которые соответствуют описанию действий, непосредственно доступных из текущего контекста. В то же время проектные операции реализуются в конкретной САПР цепочками элементарных действий пользователя. Такая особенность организации взаимодействия проектировщика с системой требует механизма пошаговой помощи, основанной на генерации цепочки действий, необходимых для выполнения запрошенной операции из текущего контекста.

Многошаговые СКП действуют подобно автомобильным навигаторам, которые формируют рекомендованный маршрут между парой заданных на карте точек через множество промежуточных пунктов и динамически изменяют его в случае отклонений. Эффект сокращения затрат времени на поиск контекстной справочной информации в этом случае проявляется тем сильнее, чем больше длина цепочки [5].

При использовании в качестве модели сценариев действий проектировщика сетей Петри задача построения цепочек действий на СП может быть решена двумя способами. Первый способ – это анализ достижимости целевого (возникшего в результате выполнения запрошенной операции) контекста из текущего. Хотя анализ достижимости является частью стандартной аксиоматики СП, он основан на применении стратегии исчерпывающего перебора и является неэффективным в вычислительном отношении. В [6] для построения цепочек действий, необходимых для выполнения запрошенной операции, предложен метод инверсии СП. Данный метод реализует ограниченный перебор, что обеспечивает меньшие, по сравнению с исчерпывающим перебором, вычислительные затраты. Для примера рассмотрим операцию экспорта объектов заранее созданной выборки в файл CDE в «CADLib Модель и Архив» – инструменте для управления BIM-проектом объектов капитального строительства и технологического оборудования крупных промышленных предприятий [1]. Обсудим запрос целевой маркировки, обусловленной срабатыванием перехода $Click\ Publish\ Catalog\ Data$. в сети сценария (рис. 3).

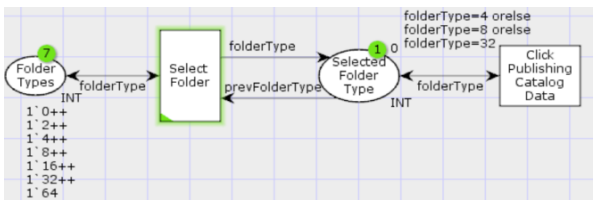


Рис. 3. Фрагмент СП «Экспорт каталога»

В случае использования стратегии исчерпывающего поиска состояний СП с учетом недетерминизма выбора маркеров и переходов

для срабатывания будут перебраны все семь возможных значений маркеров в позиции $Selected\ Folder\ Type$.

Результат поиска инвертированной СП совпадает с результатом исчерпывающего поиска состояний СП. Преимуществом инвертированной СП является то, что множество допустимых значений переменной $folderType$ в области видимости перехода $clickPublishCatalogData$ сразу ограничивается значениями 4, 8 и 32 (согласно охранному выражению). Таким образом, количество рассматриваемых вариантов сразу уменьшается с семи до трех.

При наличии в рассматриваемой для запрошенной операции подсети нескольких переходов с охранными выражениями количество ветвей, которые необходимо рассмотреть, равно произведению всех возможных вариаций, в том числе значений переменных на охранных выражениях. Таким образом, вычислительная сложность поиска инвертированной СП значительно ниже исчерпывающего перебора.

Однако аксиомы инверсии, во-первых, не являются частью стандартной модели СП, во-вторых, для раскрашенных СП, которые используются в качестве модели представления сценария, они строятся на ряде допущений, касающихся областей определения маркеров и вида выражений на дугах и переходах. В [6] первая проблема была решена путем разработки внешнего по отношению к модели СП способа описания и механизма интерпретации данных аксиом на объектно-ориентированном языке программирования C#. Но при таком подходе вновь возникает проблема «двуязычия»: с одной стороны, для описания СП используются декларации языка XML, с другой – процедурный язык для описания аксиом инверсии и зависимого от этих аксиом механизма их интерпретации. Необходимость подстраивать механизм интерпретации под изменяющееся количество и формулировки аксиом является главным недостатком данного подхода, осложняющим встраивание СКП в различные САПР.

В связи с этим целью данного исследования является обеспечение необходимой гибкости системы контекстной помощи за счет использования единой модели представления знаний о сценарии действий пользователя, аксиом инверсии, определенных на модели сценария, и унифицированного механизма интерпретации этих аксиом в контексте модели сценария.

Материалы и методы. В рамках данного исследования предлагается использовать язык логического программирования PROLOG[3] в качестве средства описания СП, так как именно логическая модель адекватно учитывает декларативную и процедурную семантику СП. Развернутая аргументация этого тезиса приведена ниже.

Для иллюстрации семантики СП рассмотрим фрагмент модели (рис. 3) с текущей разметкой со значением 32 в позиции Selected Folder Type.

1. С точки зрения декларативной семантики модель СП – граф (P, T, F, M, μ), устанавливающий отношение $F \subseteq (P \times T) \cup (T \times P)$ [4] – множество дуг, которые связывают элементы из множеств позиций P и переходов T, и определяющий функцию отображения множества маркеров на множество позиций $\mu: M \to P$ [4].

Для рассматриваемого примера:

- $P = \{Folder\ Types,$ $Selected\ Folder\ Type\}$;
- $T = \{Select\ Folder,$ $Click\ Publish\ Catalog\ Data\}$;
- $F =$
  - $\{(Folder\ Types,\ Select\ Folder),$
  - $(Select\ Folder,\ Folder\ Types),$
  - $(Select\ Folder,$ $Selected\ Folder\ Type),$
  - $(Selected\ Folder\ Type,$ $Select\ Folder),$
  - $(Selected\ Folder\ Type,$ $Click\ Publish\ Catalog\ Data),$
  - $(Click\ Publish\ Catalog\ Data,$ $Selected\ Folder\ Type)\}$;
- $M = \{0,\ 2,\ 4,\ 8,\ 16,\ 32,\ 32,\ 64\}$;
- $\mu = \{(\text{Folder Types},\ \{0,\ 2,\ 4,\ 8,$ $16,\ 32,\ 64\}),$ $(\text{Selected Folder Type},\ \{32\})\}$.

2. С точки зрения процедурной семантики СП могут быть описаны следующим образом [4]:

a) переход $t \in T$ активен, если во всех входных позициях $p \in P \mid (p, t) \in F$ есть маркеры $m \in M$, значение которых удовлетворяет выражениям на дугах и охранному выражению перехода;

b) если переход $t \in T$ активен из состояния M_1, его срабатывание переведет СП в состояние M_2. Это можно обозначить как $M_1 \xrightarrow{t} M_2$;

c) состояние M_n достижимо из состояния M_1, если существует последовательность переходов $\sigma = \{t_1 t_2 .. t_n\}$ такая, что $M_1 \xrightarrow{t_1} M_2 \xrightarrow{t_2} \dots \xrightarrow{t_n} M_n$. Иными словами $M_1 \xrightarrow{\sigma} M_n$;

d) построение всех возможных последовательностей срабатываний переходов при анализе достижимости $M_1 \xrightarrow{\sigma} M_n$ базируется на использовании стратегии исчерпывающего поиска состояний СП. В общем случае выбор перехода СП для активации не детерминирован.

Для логических языков программирования, в частности для PROLOG, также характерно сочетание декларативной и процедурной семантик.

[3] ISO/IEC 13211-1:1995 /Cor 3:2017 Information technology – Programming languages – Prolog – Part 1: General core.

Декларативная семантика логической программы (ЛП) – это описание отношений (предикатов) между объектами (термами). Поэтому структура и состояние СП – отношения между объектами конкретной предметной области – естественно описываются в виде утверждений для предикатов логического языка.

Например, фрагмент СП, представленный на рис. 3, может быть описан в PROLOG следующим образом:

Позиции (Название, Маркировка):

- $place(selectedFolderType, [32])$.
- $place(folderTypes, [0,2,4,8,16,32,64])$.

Переходы (Название, Входные дуги, Выходные дуги, Переменные в области видимости, Переменные в охранном выражении):

- $transition(selectFolder)$.
- $transition(clickPublishCatalogData)$.

Дуги (ID, Позиция, Переход, Направление, Переменная):

- $arc("ID10", selectedFolderType, clickPublishCatalogData, 1, folderType)$.
- $arc("ID11", selectedFolderType, clickPublishCatalogData, -1, folderType)$.
- $arc("ID14", folderTypes, selectFolder, -1, folderType)$.
- $arc("ID15", folderTypes, selectFolder, 1, folderType)$.
- $arc("ID16", selectedFolderType, selectFolder, -1, folderType)$.
- $arc("ID17", selectedFolderType, selectFolder, 1, prevFolderType)$.

Таким образом, отношения СП естественным образом описываются в формате утверждений PROLOG.

Процедурная семантика логической программы – это аксиомы метода резолюций, определяющие смену состояний управления (аксиомы выбора и решения вызова) и данных (аксиомы унификации) в процессе выполнения запроса к ЛП. Построенная на этих аксиомах машина вывода использует универсальную для любых предикатов логической программы стратегию исчерпывающего поиска для построения траектории управления и универсальную для любых термов логической программы процедуры связывания и освобождения данных, реализующей аксиомы унификации. Так как статус активности переходов является частным случаем состояния управления, а текущая разметка СП – это частный случай состояния связанных данных, применение универсальной стратегии исчерпывающего поиска для анализа динамики СП является обоснованным.

Например, предикаты проверки активности перехода могут быть реализованы следующим образом:

$$may\_be\_fired(Transition) :- transition(Transition, InputList), contains\_tokens(InputList).$$

Иными словами, переход активен, если по всем входным дугам ($InputList$) есть необходимые маркеры.

На основании описанной топологии СП, а также правил ее поведения возможен поиск цепочек действий для перехода из текущего состояния в запрошенное. При этом обработка будет производиться с учетом недетерминизма выбора перехода для активации и маркеров, так как недетерминизм решения вызовов – встроенный механизм пролога. Например, при срабатывании перехода $Select\ Folder$ выбор маркера из позиции $Folder\ Types$ недетерминирован, т.е. переменной $folderType$ может быть присвоено любое значение из множества $\mu(folderType)$.

Таким образом, СП естественным образом описываются средствами логического языка программирования PROLOG. Кроме того, использование логического языка для описания СП обеспечивает необходимую гибкость в реализации аксиом инверсии [7], так как аксиомы метода резолюций универсально применимы к любой логической модели. Следовательно, добавление в эту модель аксиом инверсии не повлечет за собой изменения машины вывода.

Еще одним достоинством данного подхода является возможность быстрой интеграции машины вывода в программные приложения, написанные на объектно-ориентированных языках программирования. Например, свободно распространяемый проект с открытым исходным кодом CSharpProlog[4] позволяет использовать программы на языке PROLOG в приложениях на языке c#.

Результаты. Основными результатами данного исследования являются:

1. *Предикатная модель аксиом инверсии*. Запрос на поиск цепочек переходов предполагает последовательное движение по инвертированным дугам из позиций целевой разметки в позиции текущей разметки. На каждом шаге топологического анализа возникает виртуальная разметка M_n такая, что $M_1 \xrightarrow{\sigma} M_n$, где σ – список переходов, активация которых переводит СП из текущего состояния в состояние M_n. Анализ завершается, когда целевая разметка $M_G \subseteq M_n$. Динамическая траектория выполнения запроса представляет собой И/ИЛИ дерево инверсных срабатываний (рис. 6), такое, что активация переходов снизу вверх (с учетом параллельных ветвей, связанных условием И) приводит к целевой разметке.

2. *Метод линеаризация дерева для генерации множества вариантов цепочек дей-*

[4] CSharpProlog: [библиотека классов]. – URL: https://github.com/jsakamoto/CSharpProlog/ (дата обращения: 28.12.2022).

ствий. Под линеаризацией И/ИЛИ дерева понимается отображение квазипараллельных действий (срабатываний переходов СП) в линейную структуру (цепочку).

Для сформулированных аксиом приняты следующие ограничения:

1) СП – плоская. Представление иерархической СП в предикатной форме необоснованно усложнило бы формирование логической программы. При инверсном анализе подстановочные переходы могут быть рассмотрены как обычные. При необходимости внутренняя структура такого перехода может быть рассмотрена отдельно;

2) в охранных выражениях допустимо использовать операторы «=», «И», «ИЛИ»;

3) в выражениях на дугах из позиции в переход допускается только одна переменная для типизированных позиций.

Ограничения второе и третье связаны с тем, что СКП не ограничивает действия пользователя, а только регистрирует их выполнение.

Аксиомы инверсии функционально делятся на 3 группы. Аксиомы первой группы определяют условия получения маркера из позиции. Аксиомы второй группы определяют условия срабатывания перехода, ссылаясь при этом на аксиомы первой группы. Аксиомы третьей группы определяют варианты связывания переменных на дугах СП в процессе виртуальной разметки позиций. При записи этих аксиом будем использовать следующие обозначения:

unit – черно-белый маркер;

any – любое типизированное значение;

Var – название переменной;

empty – пустое выражение на дуге;

[] – пустой список;

[П|С] – разделение списка на первый элемент и остальные.

Аксиомы первой группы представлены следующими утверждениями:

1. Черно-белый маркер может быть взят из позиции P, если маркировка этой позиции не пуста:
$$\text{получить\_маркер}(P, unit) :- \text{маркировка}(P, M), M \neq \emptyset.$$
2. Типизированный маркер может быть взят из позиции P, если маркировка этой позиции содержит маркер с запрошенным значением:
$$\text{получить\_маркер}(P, Value) :- \text{маркировка}(P, M), Value \in M.$$
3. Типизированный маркер может быть взят из позиции P, если по одной из входных дуг в нее может быть доставлен маркер с запрошенным значением:
$$\text{получить\_маркер}(P, Value) :- \text{дуга}(T, P, Var), \text{связать}(T, Var, Value), \text{активировать\_переход}(T).$$
4. Типизированный маркер может быть взят из позиции P, если маркировка этой позиции не пуста:
$$\text{получить\_маркер}(P, any) :- \text{маркировка}(P, M), M \neq \emptyset.$$
5. Типизированный маркер может быть взят из позиции P, если он может быть помещен туда в результате срабатывания перехода:
$$\text{получить\_маркер}(P, any) :- \text{дуга}(T, P, Var), \text{связать}(T, Var, any), \text{активировать\_переход}(T).$$

Аксиомы связывания переменных в области видимости перехода представлены следующими утверждениями:

1. Переменные в области определения перехода T могут быть связаны со значениями, если каждая переменная в области видимости может быть связана со значением:
$$\text{связать}(T, Var, Value) :- \text{переход}(T, Vars, GuardVars), \text{связать\_переменные}(T, Vars, VarValues, GuardVars, [(Var, Value)]).$$
2. Список переменных может быть связан со значениями, если первая переменная участвует в охранном выражении, запрошенное для нее значение удовлетворяет охранному выражению и оставшийся список переменных может быть связан со значениями:
$$\text{связать\_переменные}(T, [Var|Vars], VarValues, GuardVars, RequestedValues) :- Var \in GuardVars, (Var, Value) \in RequestedValues, \text{охранное\_выражение}(T, Var, Value), \text{вставить}((Var, Value), VarValues), \text{связать\_переменные}(T, Vars, GuardVars, RequestedValues).$$
3. Переменные списка могут быть связаны значениями, если первая переменная может быть связана значением в соответствии с охранным выражением и переменные в оставшейся части списка также могут быть связаны со значениями:
$$\text{связать\_переменные}(T, [Var|Vars], GuardVars, RequestedValues) :- Var \in GuardVars, \text{охранное\_выражение}(T, Var, Value), \text{вставить}((Var, Value), VarValues), \text{связать\_переменные}(T, Vars, GuardVars, RequestedValues).$$
4. Переменные списка могут быть связаны значениями, если первая переменная может быть связана запрошенным значением и переменные в оставшейся части списка также могут быть связаны со значениями:

связать_переменные$(T, [Var|Vars],$
$GuardVars,\ RequestedValues) :-$
$(Var, Value) \in RequestedValues,$
вставить$\big((Var, Value), VarValues\big),$
связать_переменные$(T, Vars,$
$GuardVars,$
$RequestedValues).$

5. Список переменных может быть связан со значениями, если для первой переменной не запрошено значение, тогда она связывается со значением any и оставшийся список переменных может быть связан со значениями:

связать_переменные$(T, [Var|Vars],$
$GuardVars,\ RequestedValues) :-$
вставить$\big((Var, any), VarValues\big),$
связать_переменные$(T, Vars,$
$GuardVars,$
$RequestedValues).$

Аксиомы активации переходов представлены следующими утверждениями:

1. Переход T может быть активирован, если можно взять маркеры из всех входных позиций с учетом запрошенных значений переменных:

активировать_переход$(T, VarValues) :-$
переход$(T, InputArcs),$
получить_маркеры_по_дугам$(InputArcs,$
$VarValues).$

2. По дугам могут быть взяты маркеры с запрошенными значениями переменных, если:
   1) первая дуга связана с черно-белой позицией и из нее можно получить черно-белый маркер;
   2) по остальным дугам могут быть взяты маркеры:

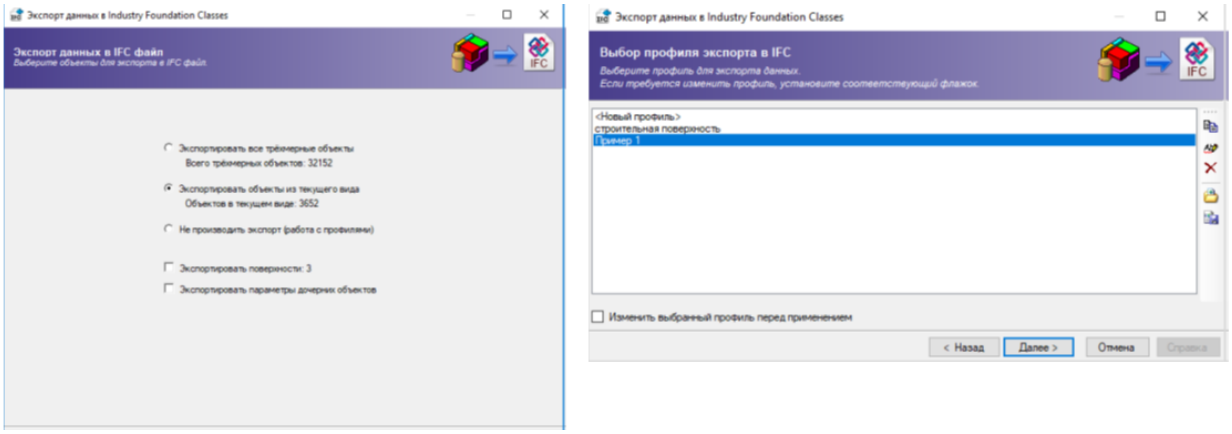
получить_маркеры_по_дугам$([Arc|Arcs],$
$VarValues) :-$
дуга$(Arc, P, T, empty),$
получить_маркер$(P, unit),$
получить_маркеры_по_дугам$(Arcs,$
$VarValues).$

3. По дугам могут быть взяты маркеры с запрошенными значениями переменных, если:
   1) первая дуга связана с типизированной позицией и из нее можно получить маркер со значением в соответствии с выражением на дуге;
   2) по остальным дугам могут быть взяты маркеры:

получить_маркеры_по_дугам$([Arc|Arcs],$
$VarValues) :-$
дуга$(Arc, P, T, Var),$
$(Var, Value) \in VarVlues,$
получить_маркер$(P, Value),$
получить_маркеры_по_дугам$(Arcs,$
$VarValues).$

В процессе выполнения запроса на построение цепочек рекомендованных действий формируется динамическая траектория выполнения запроса в виде дерева, дуги которого могут быть объединены связками «И»–«ИЛИ». Связка «И» отражает условие наличия маркеров во всех входных позициях активного перехода. Связка «ИЛИ» отражает альтернативные условия маркировки позиции в результате срабатывания любого из переходов, для которых она является выходной. В МиА у пользователя есть возможность экспортировать объекты в формате IFC. В процессе выполнения цепочки действий, полученной в результате запроса КП «Экспорт в IFC», пользователь может выбрать объекты для экспорта (рис. 4,а) или осуществить работу с профилями экспорта (редактирование, добавление и т.д.) (рис. 4,б). И/ИЛИ дерево для соответствующей СП (рис. 5) приведено на рис. 6.

Для удобства сопоставления СП и построенного на нем дерева И/ИЛИ для запроса «Экспорт в IFC» его узлы изображены в соответствии со стандартом визуализации СП – позиции представлены в виде овалов, переходы – в виде прямоугольников. Ветви, объединенные условием «И», соединены дугой у корня.



а) б)

Рис. 4. Экспорт в IFC: а – выбор параметра экспорта; б – работа с профилями экспорта

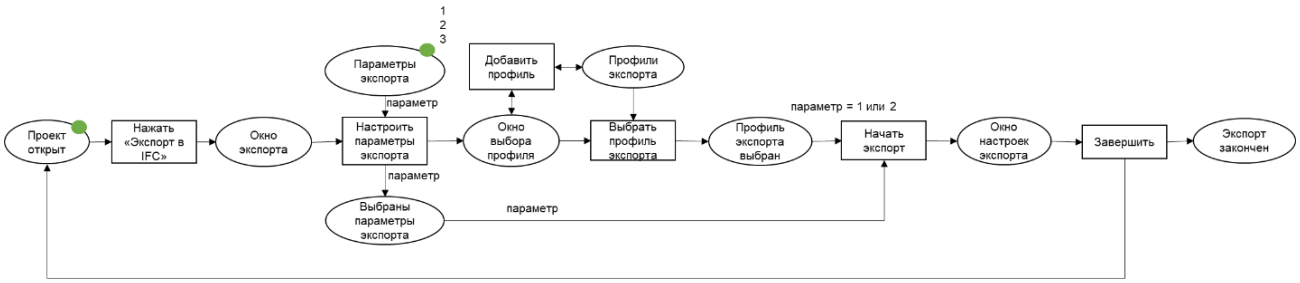


Рис. 5. СП «Экспорт в IFC»

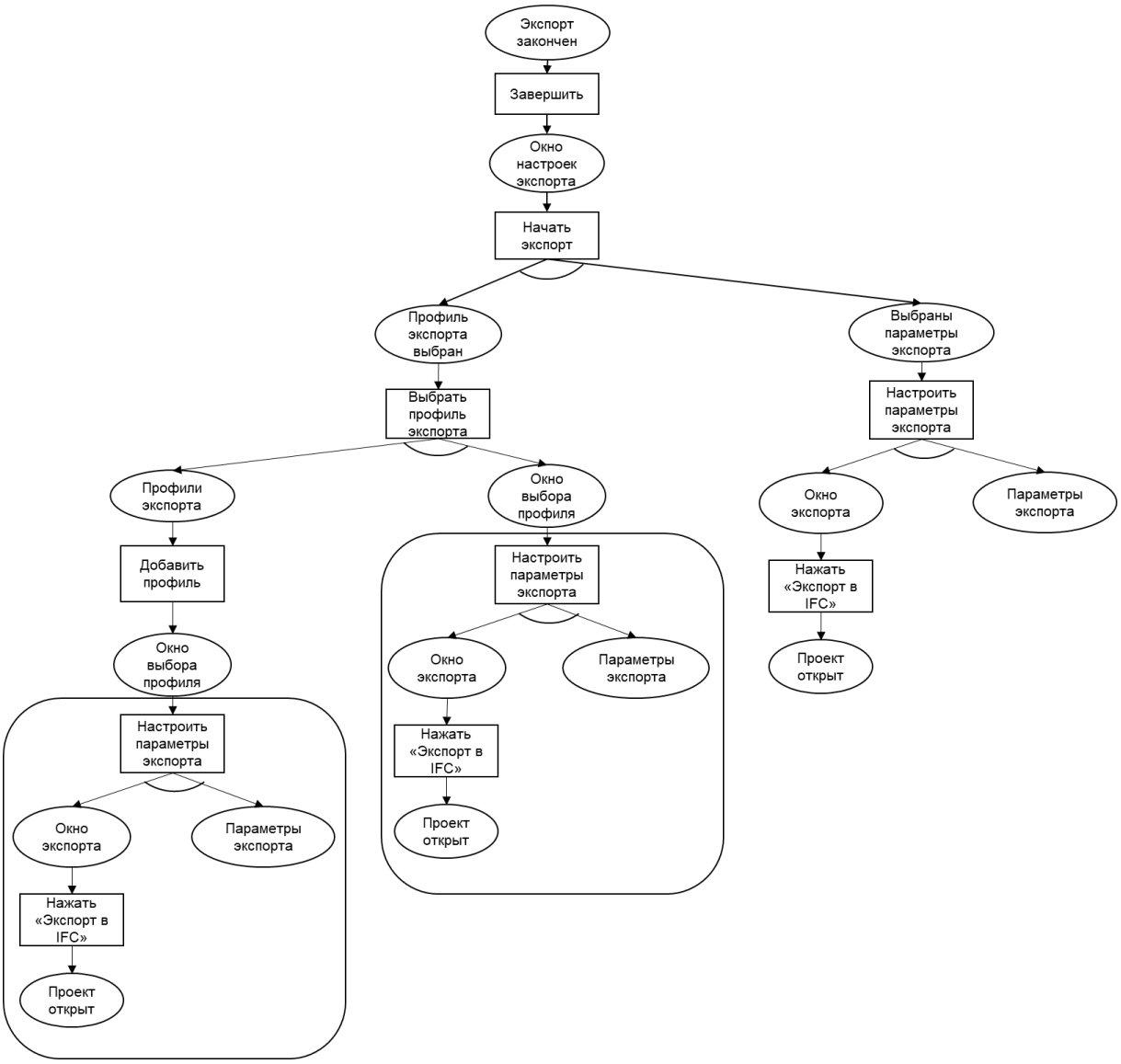


Рис. 6. И/ИЛИ дерево

При построении данного дерева в момент рассмотрения перехода «Выбрать профиль экспорта» две подветви объединяются с учетом того, что множество переходов в одной является подмножеством переходов в другой (выделены скругленными прямоугольниками). Результат объединения представлен на рис. 7,а. Далее объединяются две ветви при рассмотрении перехода «Начать экспорт», в результате чего образуется одна цепочка действий (рис. 7,б).

Так как недетерминированный выбор перехода для активации осуществляется за счет внутренних механизмов PROLOG, то осуществлять отдельно необходимо только объединение квазипараллельных веток. При описании аксиом попарного объединения цепочек переходов использованы следующие обозначения:

σ_1 – список переходов из первой подветви;

σ_1 – список переходов из второй подветви;

σ – объединенная цепочка переходов.

Ниже приведены аксиомы попарного объединения:

$\sigma = \sigma_2$, если $\sigma_1 = \emptyset$;

$\sigma = \sigma_1$, если $\sigma_2 = \emptyset$;

$\sigma = \sigma_2$ или $\sigma = \sigma_1 \cup \sigma_2$ или $\sigma = \sigma_2 \cup \sigma_1$, если $\sigma_1 \subseteq \sigma_2$;

$\sigma = \sigma_1$ или $\sigma = \sigma_1 \cup \sigma_2$ или $\sigma = \sigma_2 \cup \sigma_1$, если $\sigma_2 \subseteq \sigma_1$;

$\sigma = \sigma_1 \cup \sigma_2$ или $\sigma = \sigma_2 \cup \sigma_1$, если $\sigma_2 \nsubseteq \sigma_1$ и $\sigma_1 \nsubseteq \sigma_2$.

С помощью CSharpProlog возможна организация СКП, представленная на рис. 8. По запросу способа выполнения операции «Экспорт в IFC» СКП генерирует альтернативные цепочки возможных действий. Каждый из элементов цепочки представляет собой ссылку на описание способа выполнения элементарного действия в файле справки МиА. В данном примере использованы отекстовки из файла конфигурации, которые могут изменяться в зависимости от значения переменной при активации перехода.

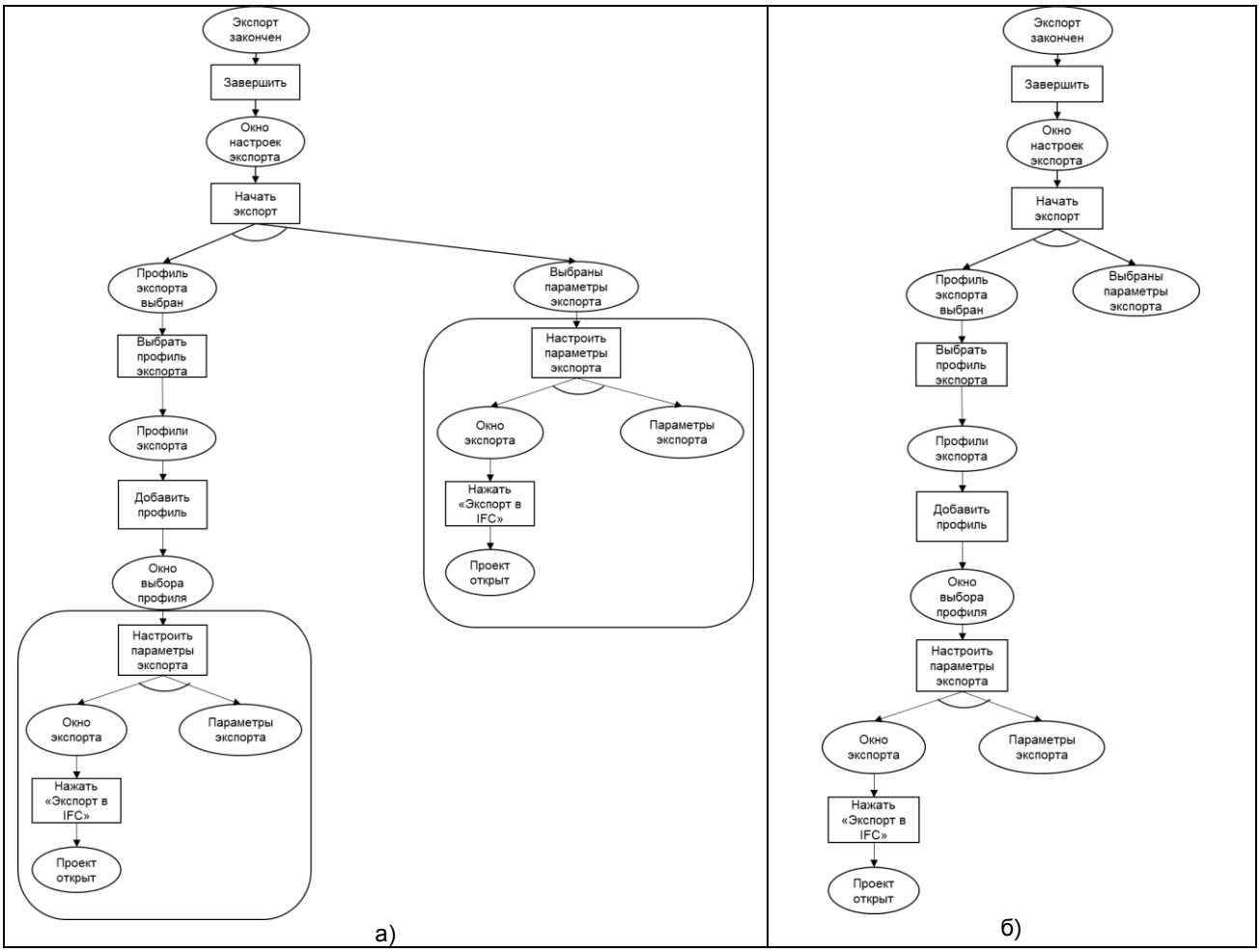


Рис. 7. Этапы объединения ветвей И/ИЛИ дерева: а – И/ИЛИ дерево после объединения ветвей; б – финальная цепочка действий

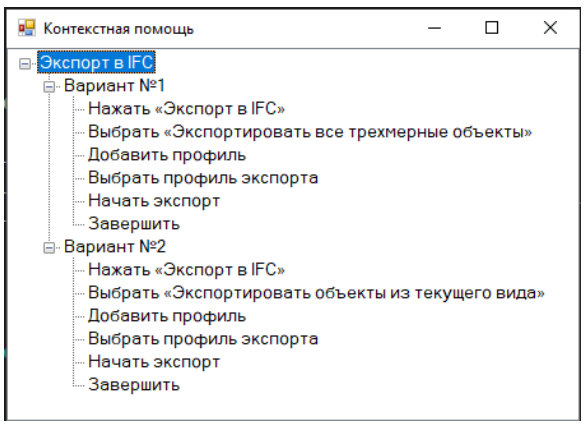


Рис. 8. Окно контекстной помощи

Двойной клик по узлу представленного дерева приведет к открытию файла справки с нужным позиционированием.

Выводы. Экспериментальное исследование программной реализации предложенной единой модели представления знаний о сценарии действий пользователя, аксиом инверсии, определенных на модели сценария, и унифицированного механизма интерпретации этих аксиом в контексте модели сценария обеспечивает возможность организации системы пошаговой контекстной помощи (СПКП). Данный подход позволяет добавлять и редактировать аксиомы инверсии без изменения машины вывода. Таким образом, цель исследования –

обеспечение необходимой гибкости системы контекстной помощи – достигнута.

Достоверность результатов исследования подтверждается как обоснованным выбором апробированных инструментов (СП для представления моделей сценариев действий пользователя, логический язык программирования PROLOG для предикатной формы представления СП и аксиом инверсии), так и совпадением результата их применения с ожидаемым.

Полученные результаты могут быть использованы в процессе разработки и поддержки программных продуктов, в частности САПР, в качестве ядра СПКП. Данные приложения должны информировать СПКП о действиях пользователя, в результате чего будет обеспечено актуальное состояние СП. СПКП реализована в виде динамически подключаемой библиотеки (.dll) и позволяет использовать как встроенный пользовательский оконный интерфейс, так и методы генерации цепочек действий по запросу пользователя для отображения в любом другом виде. Программный интерфейс данной библиотеки предоставляет методы информирования о действиях пользователя (для активации переходов), инициализации начальной маркировки и запроса пошаговой контекстной помощи. Использование логической программы в качестве ядра системы обеспечено библиотекой CSharpProlog.

Дальнейшим направлением научного исследования является расширение аксиоматики инверсии за счет поддержки более широкого спектра маркеров, сложных арифметических и логических выражений на дугах и в охранных выражениях СП. Данные изменения позволят более полно использовать механизмы СП и упростить их структуру за счет более полного использования ограничений в виде охранных выражений и выражений на дугах.

Список литературы

1. **CADLib** Модель и Архив // CADmaster. – 2020. – № 2(93). – С. 70–74.

2. **Crowley R., Medvedeva O.** An intelligent tutoring system for visual classification problem solving // Artificial Intelligence in Medicine. – 2006. – No. 6. – P. 85–117.

3. **Котов В.Е.** Сети Петри. – М.: Наука, 1984. – 160 с.

4. **Van der Aalst W.M.P.** The application of Petri nets to workflow management // Journal of circuits, systems, and computers. – 1998. – Vol. 8, No. 01. – P. 21–66.

5. **Пантелеев Е.Р., Мукучян А.А.** Модель и метод построения контекстной помощи для отработки навыков оператора энергоблока на тренажере // Вестник ИГЭУ. – 2021. – Вып. 3. – С. 66–75. DOI: 10.17588/2072-2672.2021.3.066-075.

6. **Метод** формирования контекстной помощи пользователю компьютерного приложения в процессе решения прикладной задачи / Е.Р. Пантелеев, А.А. Мукучян, М.А. Кузнецов, А.Л. Алыкова // Вестник ИГЭУ. – 2020. – Вып. 5. – С. 64–76. DOI: 10.17588/2072-2672.2020.5.064-076.

References

1. CADLib Model' i Arkhiv [Model and Archive]. *CADmaster*, 2020, № 2(93), pp. 70–74.

2. Crowley, R., Medvedeva, O. An intelligent tutoring system for visual classification problem solving. *Artificial Intelligence in Medicine*, 2006, no. 6, pp. 85–117.

3. Kotov, V.E. *Seti Petri* [Petri nets]. Moscow: Nauka, 1984. 160 p.

4. Van der Aalst, W.M.P. The application of Petri nets to workflow management. *Journal of circuits, systems, and computers,* 1998, issue 8, no. 01, pp. 21–66.

5. Panteleev, E.R., Mukuchyan, A.A. Model' i metod postroeniya kontekstnoy pomoshchi dlya otrabotki navykov operatora energobloka na trenazhere [Model and method of contextual help to develop skills of a power unit operator using training simulator]. *Vestnik IGEU*, 2021, issue 3, pp. 66–75. DOI: 10.17588/2072-2672.2021.3.066-075.

6. Panteleev, E.R., Mukuchyan, A.A., Kuznetsov, M.A., Alykova, A.L. Metod formirovaniya kontekstnoy pomoshchi pol'zovatelyu komp'yuternogo prilozheniya v protsesse resheniya prikladnoy zadachi [Method of context-dependent assistance for software user solving an applied task]. *Vestnik IGEU*, 2020, issue 5, pp. 64–76, DOI: 10.17588/2072-2672.2020.5.064-076.