УДК 621.398
DOI: 10.15827/0236-235X.138.145-152
Дата подачи статьи: 12.04.22, после доработки: 22.04.22
2022. Т. 35. № 2. С. 145–152

Разработка прототипа решателя для расширенных шаговых теорий логики высказываний

И.Б. Фоминых [1], *д.т.н., профессор, fominykhIB@appmat.ru*
Н.П. Алексеев [1], *ст. преподаватель, AlekseevNP@mpei.ru*
Н.А. Гулякина [2], *к.ф.-м.н., доцент, зав. лабораторией, guliakina@bsuir.by*
К.С. Кравченко [1], *магистрант, KravchenkoKS@mpei.ru*
М.В. Фомина [1], *к.т.н., доцент, FominaMV@mpei.ru*

[1] *Национальный исследовательский университет «МЭИ», г. Москва, 111250, Россия*
[2] *Белорусский государственный университет информатики и радиоэлектроники (БГУИР), г. Минск, 220013, Беларусь*

В настоящее время проводятся активные исследования возможностей использования неклассических логик в моделировании рассуждений когнитивного агента.

В статье рассматривается проблема разработки и реализации прототипа решателя расширенных шаговых теорий в случае, когда решения по управлению сложным техническим объектом принимаются в условиях жестких временных ограничений. Рассматривается логическая система, основанная на использовании шаговых теорий с двумя видами отрицания, – система расширенных шаговых теорий. Использование двух видов отрицания позволяет выводить как истинные факты, так и факты-убеждения, что важно при моделировании рассуждений человека.

Основное внимание уделяется вопросу организации процедуры вывода на основе использования неклассических логик в моделировании рассуждений когнитивного агента.

Приводятся основные этапы разработки прототипа расширенных шаговых теорий с использованием литералов логики высказываний. Для каждого компонента решателя описаны его функции, задачи, входные и выходные данные. Обоснован выбор системы вывода clingo, поддерживающей формирование расширенных логических программ Answer Set Programming (ASP) как инструмента реализации решателя.

Приведены алгоритмы трансляции расширенных шаговых теорий в логическую программу, соответствующую синтаксису ASP. При организации логического вывода использован алгоритм циклической обработки множеств убеждений расширенных шаговых теорий в среде clingo. Основные этапы работы этого алгоритма рассмотрены на примере, где разбираются этапы работы решателя и приводятся результаты, представленные в синтаксисе clingo. Пример работы решателя демонстрирует основные особенности расширенных шаговых теорий в задачах жесткого реального времени, такие как отказ от логического всеведения, самопознание и темпоральная чувствительность.

В дальнейшем планируется рассмотреть применимость созданного решателя к более сложной формальной системе – логике предикатов первого порядка.

Ключевые слова: *расширенная шаговая теория, решатель, активная логика, ограничения по времени, логическое программирование.*

При моделировании рассуждений, связанных с обработкой временных зависимостей, в частности, в многоагентных системах [1], важным является понятие когнитивного агента. Подходы к организации логических рассуждений агента, происходящих во времени, рассматривались в работах [2–6]. Они представляют собой рассуждения о времени, то есть внесение в правила вывода темпоральных литералов и правил их обработки. В данной работе рассматривается другой подход к рассуждениям когнитивного агента – рассуждения во времени, предполагающий соотнесение результатов логического вывода агента с моментами времени, в которые они были получены.

Базовыми для концепции рассуждений во времени можно назвать формализмы *активной логики* (АЛ) – концептуальной системы, объединяющей в себе ряд формализмов рассуждений, происходящих во времени, которые рассматриваются как протекающий во времени процесс [7]. Формализмы, отвечающие принципам АЛ, подходят для решения задач управления рассуждениями в условиях жестких временных ограничений [8], когда превышение допустимого количества времени на решение, называемого порогом, может приводить к катастрофическим последствиям. Рассуждения, удовлетворяющие данной концепции, обладают следующими важными свойствами: во-

первых, результаты рассуждений могут быть соотнесены с моментами времени, в которые они были получены, и когнитивный агент способен это осознавать (свойство темпоральной чувствительности), и, во-вторых, эти рассуждения толерантны к возникающим в процессе рассуждения противоречиям (паранепротиворечивость). В работе [9] предложена паранепротиворечивая аргументационная семантика для формализма *шаговых теорий* (ШТ), объединяющего в себе принципы АЛ и логического программирования. В дальнейшем был разработан представленный в [10] более общий вариант формализма, содержащий два вида отрицания – *расширенные шаговые теории* (РШТ). В работе [11] было выполнено соотнесение РШТ и расширенных логических программ, основанных на семантике ответных множеств (Answer Set Programming, ASP), которые, как известно, являются частными случаями теорий с умолчаниями Р. Рейтера.

В данной работе представлено дальнейшее развитие формализма РШТ с использованием ASP. Предложена архитектура решателя для ШТ, содержащих в правилах лишь пропозициональные литералы. Разработан действующий прототип решателя, использующий в качестве модулей существующие системы вывода ASP.

РШТ как формализм, отвечающий принципам АЛ

Для моделирования рассуждений в системах жесткого реального времени существует необходимость использования темпорально чувствительных логик, так как в таких системах пренебрежение временем реакции на изменения в окружающей среде неприемлемо. Кроме того, для задач управления сложными объектами в режиме жесткого реального времени характерно поступление новой, зачастую противоречивой информации, которую необходимо правильно обработать, чтобы время, отведенное на решение этих задач, не превысило допустимый порог. АЛ позволяет рассматривать рассуждения, происходящие во времени, не как статичную последовательность утверждений, а как протекающий во времени процесс. АЛ обладает несколькими интересными с точки зрения решаемой задачи особенностями: отказ от логического всеведения, самопознание и темпоральная чувствительность [7].

Одной из систем АЛ являются ШТ. Множество правил ШТ разбиты на два подмножества – строгих и правдоподобных правил. В дальнейшем будем предполагать, что в теории присутствуют только правдоподобные правила вида

$$N\colon a_1 \wedge a_2 \wedge \ldots \wedge a_m \Rightarrow b, \quad (1)$$

где N – имя правила; b – пропозициональный литерал; a_i – пропозициональные литералы или литералы языка логики первого порядка вида later(j) или ¬later(j) (j – натуральное число, сопоставленное определенному моменту времени). Эти правила означают, что если выполняется формула $a_1 \wedge a_2 \ldots \wedge a_m$ и на данном шаге неизвестно, правдоподобна ли формула $\neg b$, то можно предположить, что формула b выполняется. Таким образом, система ШТ может рассматриваться как вариант системы активной логики, которая полностью основана на правилах и отвечает принципу логического программирования, в соответствии с которым моделями формул являются множества литералов, а не более сложные структуры в стиле Крипке.

Часы ШТ представляют собой конечную монотонно возрастающую подпоследовательность последовательности натуральных чисел, обозначенную далее *Ck*. Члены данной подпоследовательности характеризуют длительность последовательно выполняемых дедуктивных циклов, определяющих процесс рассуждения во всех системах АЛ [11].

Назовем ШТ пару $T = (R, Ck)$, где R – конечное множество правил вида (1); Ck – конечная или бесконечная строго возрастающая подпоследовательность глобальных часов, члены которой являются моментами времени завершения дедуктивных циклов, называющаяся часами прогона модели.

Синтаксис расширенной ШТ отличается от синтаксиса других ШТ наличием в его алфавите языка правил оператора субъективного отрицания not^t. Таким образом, правила будут иметь следующий вид:

$$N\colon a_1 \wedge a_2 \wedge \ldots \wedge a_m \wedge \text{not}^t c_1 \wedge \text{not}^t c_2 \wedge \ldots \wedge \text{not}^t c_n \Rightarrow b, \quad (2)$$

где N – имя правила; b – пропозициональный литерал; $a_1, \ldots, a_m$ – пропозициональные литералы или литералы языка логики первого порядка вида later(j) или ¬later(j) (j – натуральное число); $c_1, \ldots, c_n$ – пропозициональные литералы; not^t – оператор субъективного отрицания. В отличие от оператора not$x$, обозначающего безуспешность попытки вывести литерал x средствами данной логической программы, выражение not$^t x$ в антецеденте правила РШТ означает, что агент не успел вывести литерал к текущему моменту времени.

Правила РШТ выражают принцип негативной интроспекции в следующей интерпретации: если выполняется формула $a_1 \wedge a_2 \wedge \ldots \wedge a_m$ и на данном шаге вывода неизвестно, выполняется ли формула $c_1 \wedge c_2 \ldots \wedge c_n$, то допустимо предположить, что выполняется формула b. При этом, как уже было сказано, $-b$ означает литерал, являющийся дополнением до контрарной пары для литерала b.

Заметим, что между правилами РШТ и умолчаний в теориях с умолчаниями Р. Рейтера имеется внешнее сходство. Соотношение РШТ АЛ и расширенных логических программ, отвечающих семантике ответных множеств (как известно, являющихся частными случаями теорий с умолчаниями Р. Рейтера), рассмотрено в работе [11].

Логическое программирование ASP

Современным подходом к декларативному логическому программированию является ASP. Подход строится на понятии устойчивых моделей и семантике ответных множеств.

Логическая программа над набором атомов A представляет собой конечный набор правил r следующего вида: $a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n$, где $0 \le m \le n$ и каждый $a_i \in A$ является атомом для $0 \le i \le n$. Литералом называют атом a_i или его отрицание по умолчанию $\sim a_i$. Примем, что отрицание по умолчанию – это отсутствие информации об a_i (формальная запись $a_i \notin X$) в отличие от классического отрицания $\neg a_i$ (формально $\neg a_i \notin X$), где X есть некоторая интерпретация.

Для каждого подобного правила r пусть

$head(r) = a_0$ – голова правила;

$body(r) = \{a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n\}$ – тело правила.

Тогда r интуитивно читается как $head(r)$ – истина, если выполняется $body(r)$. В случае, если тело правила пусто, r назовем фактом.

Представим в качестве простого примера логическую программу P_1:

$$P_1 = \begin{cases} a, \\ c \leftarrow \sim b, \sim d, \\ d \leftarrow \sim a, \sim c. \end{cases}$$

Имеем множество литералов X, тогда пусть $X^+ = \{p \in A | \, p \in X\}$ и $X^- = \{a \in A | \sim a \in X\}$. Теперь получим $body(r)^+ = \{a_1, \ldots, a_m\}$ и $body(r)^- = \{a_{m+1}, \ldots, a_n\}$.

Назовем правило r положительным, если $body(r)^- = \varnothing$. Соответственно, и логическая программа называется положительной, если все правила положительны. Набор атомов логической программы P обозначим как $atom(P)$, а набор тел всех правил программы – $body(P)$. Набор тел правил программы, имеющих схожую голову a, обозначим как $body_p(a) = \{body(r) | r \in P, head(r) = a\}$.

Множество атомов $X \subseteq A$ является моделью программы P, если $head(r) \in X$ всякий раз, когда $body(r)^+ \subseteq X$ и $body(r)^- \cap X = \varnothing$ для каждого $r \in P$. Таким образом, программа P_1 имеет шесть моделей, среди которых выделим $\{a, c\}$ и $\{a, b, c, d\}$. Модель называют минимальной, если никакое подмножество этой модели не является моделью программы [12].

В соответствии с семантикой устойчивых моделей, принятой в ASP [12], редукт P^X программы P на множестве X определим как $P^X = \{head(r) \leftarrow body(r)^+ | r \in P, body(r)^- \cap X = \varnothing\}$, где P^X является положительной программой, поэтому она имеет минимальную модель. Тогда X называют устойчивой моделью программы P, если X является минимальной моделью P^X.

Проверим полученные модели программы P_1 на устойчивость:

X	${P_1}^X$	минимальная модель ${P_1}^X$
$\{a, c\}$	${P_1}^{\{a, c\}} = \{a \leftarrow, c \leftarrow\}$	$\{a, c\}$
$\{a, b, c, d\}$	${P_1}^{\{a, b, c, d\}} = \{a \leftarrow\}$	$\{a\}$

Видно, что $\{a, c\}$ в отличие от $\{a, b, c, d\}$ является устойчивой моделью P_1.

На практике редукт P^X программы P на множестве X можно получить следующим образом:

– из программы удаляют все правила, содержащие в теле отрицание по умолчанию $\sim a$ для $a \in X$;

– удаляют также все литералы вида $\sim a$ из тел оставшихся правил.

Возвращаясь к примеру, заметим, что все атомы в модели $\{a, c\}$ устойчивы к применению правил P_1, в то время как для модели $\{a, b, c, d\}$ истинность атомов b, c и d не может быть установлена. В таком случае можно утверждать, что каждый атом устойчивой модели доказуем правилами из P. Таким образом, каждый отрицательный литерал должен быть истинным, тогда как положительным литералам достаточно быть доказуемыми. Из вышеизложенного можно заключить, что устойчивые модели формируются из голов логической про-

граммы, а значит, {*a, b, c, d*} никак не может быть устойчивой моделью программы P_1. Таким образом, в семантике ASP шагом вывода будем называть построение устойчивой модели по имеющемуся на каждом шаге множеству убеждений.

Существуют несколько активно развивающихся проектов программной реализации ASP, среди них наиболее современной системой вывода для ASP-программ можно назвать систему вывода clingo [13]. Она совмещает в себе граундер (программный компонент, который отвечает за получение пропозиционального аналога логической программы, содержащей литералы логики первого порядка) и решатель.

Темпоральности РШТ при использовании системы вывода clingo можно достигнуть итеративной обработкой правил (в соответствии с понятием шага применительно к изменению моментов времени на часах *Ck*). Основная сложность данного процесса заключается в обновлении множества известных фактов, контроле непротиворечивости и управлении правилами с условиями вида later(*t*).

Проектирование решателя

Процесс вывода в РШТ можно разделить на следующие основные этапы (рис. 1):

– анализ входной теории, разбор на составляющие подобъекты и создание объекта-теории для последующей обработки системой;

– трансляция входной теории в логическую программу ASP (в синтаксисе clingo);

– обработка логической программы при помощи системы вывода clingo;

– получение результатов и дополнение существующего множества известных убеждений;

– вывод полученного множества убеждений.

Первые четыре этапа являются повторяющимися и определяются часами прогона *Ck* обрабатываемой расширенной шаговой теории *T*. Последний этап выполняется в конце работы решателя.

На вход решателя поступает текстовый файл, содержащий описание РШТ в соответствии с синтаксисом, определенным анализатором.

Архитектура решателя предполагает реализацию перечисленных далее компонентов.

Анализатор является лексически-синтаксическим, проверяет корректность синтаксиса РШТ и создает новый объект ШТ в системе вывода.

Транслятор обрабатывает объект ШТ и формирует из него логическую ASP-программу, отвечающую синтаксису решателя clingo.

Clingo используется в качестве подсистемы, вызываемой решателем для обработки имеющихся фактов и правил с целью получения нового множества убеждений на конкретном шаге вывода.

Модификатор РШТ выполняет корректировку множества известных убеждений в соответствии с новыми результатами, а также выполняет фильтрацию правил (отбрасывает правила, гарантированно не задействованные на текущем шаге, в соответствии с часами прогона модели, удаляет правила с субъективным отрицанием уже выведенного на предыдущих этапах моделирования факта в теле).

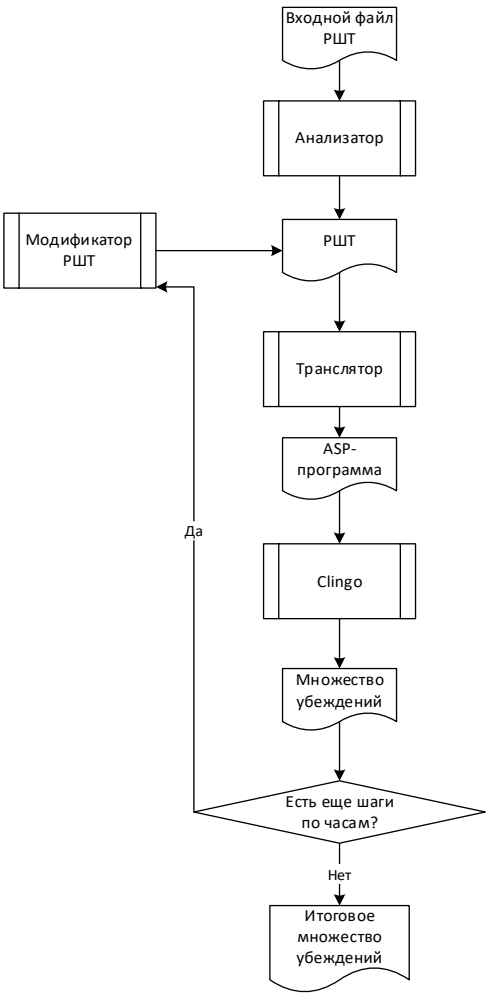

Рис. 1. Процесс вывода итогового множества убеждений

Fig. 1. The process of deriving the final set of beliefs

Представим пример входных данных, написанных на языке, заданном грамматикой анализатора (каждое правило вида *a =>b* соответствует тому факту, что из *a* выводимо *b*):

```
ТЗ: {
    R: {
        =>a,
        a =>c,
        c =>d,
        d =>f,
        !later(10) ^ d => passed,
        later(13) ^ not[t](passed) => failed
    };
        Ck = [0, 1, 3, 5, 7, 11]
}
```

Трансляция РШТ

В данный момент в логической программе присутствуют следующие виды объектов:

– have_facts(*X*): факты, известные решателю в данный момент времени (аксиомы или полученные на предыдущих шагах разбора теории);

– possible_facts(*Y*): антецеденты правил, добавленных в процессе трансляции ШТ в логическую программу;

– conj(*X*, *Y*): отношение вывода из существования объекта *X* объекта *Y*.

Модифицирующими правилами логической программы являются правила, определяющие состав ответного множества. В данный момент в решателе используется лишь правило вида res(*Y*) :- have_fact(*X*), possible_fact(*Y*), conj(*X*, *Y*).

Это правило отвечает за проверку, можно ли достигнуть тела правила *a => b* на текущем шаге разбора шаговой теории.

Обработка РШТ

При обработке РШТ начальное время на часах выставляется на третью позицию, так как в нулевой момент времени мы не знаем ничего, а при *t* = *Ck*[1] знаем только аксиомы. Потребность в обработке правил возникает только при *t* = *Ck*[2].

Если в правиле присутствует литерал later(t_i) и $t_{cur} > t_i$, то добавляем антецедент *A* в множество possible_facts, формируем из оставшихся литералов правило вида conj(*X*, *A*) и записываем его в файл.

Если в правиле присутствует литерал !later(t_i) и $t_{cur} < t_i$, то добавляем антецедент *A* в множество possible_facts, формируем из оставшихся литералов правило вида conj(*X*, *A*) и записываем его в файл.

Если в правиле присутствует литерал not[*t*](fact), то добавляем антецедент *A* в множество possible_facts, формируем из оставшихся литералов правило вида conj(n_fact, *A*) и записываем его в файл. Записываем в логическую программу множество possible_facts в виде possible_facts(*a*; *b*; …).

Демонстрация работы решателя

Приведем листинг работы решателя для РШТ с именем ТЗ, описанной ранее:

```
Запускаем анализатор
Разбор правил успешно завершен
Разбор таймера успешно завершен
Синтаксический анализ успешно завершен,
расширенная теория ТЗ получена
Запущен процесс получения 'решения'
**************************************
Запущена новая итерация.

Текущее время: 3
Файл логической программы создан
Запускаю clingo
Результат получен
Добавлен новый факт: c
**************************************
**************************************
Запущена новая итерация.

Текущее время: 5
Файл логической программы создан
Запускаю clingo
Результат получен
Добавлен новый факт: d
**************************************
**************************************
Запущена новая итерация.

Текущее время: 7
Файл логической программы создан
Запускаю clingo
Результат получен
Добавлен новый факт: f
Добавлен новый факт: passed
**************************************
**************************************
Запущена новая итерация.

Текущее время: 11
Файл логической программы создан
Запускаю clingo
Результат получен
**************************************
Моделирование завершено
Итоговое множество: ['a', 'c', 'd', 'f',
'passed']
```

На листинге показаны этапы анализа и трансляции в *lp*-файл, дальнейшие шаги за-

```
have_fact(a).
have_fact().
possible_fact(c; d; f; passed).

conj(a, c).
conj(c, d).
conj(d, f).
conj(d, passed).

res(Y) :- have_fact(X), possible_fact(Y), conj(X, Y).

#show res/1.
```

```
have_fact(a; c; d; f; passed).
have_fact().
possible_fact(c; d; f; passed).

conj(a, c).
conj(c, d).
conj(d, f).
conj(d, passed).

res(Y) :- have_fact(X), possible_fact(Y), conj(X, Y).

#show res/1.
```

Рис. 2. Содержимое lp-файла

Fig. 2. The lp file contents

пуска clingo и модификации множества убеждений в процессе решения. Из листинга видно, какие факты и на каком шаге часов были получены. Пример демонстрирует обработку литералов later в правилах РШТ в соответствии с часами модели. Поскольку литерал *d* был успешно выведен на шаге часов 5, на шаге часов 7 стало выполнимо правило !later(10) ^ *d* => *passed*, приводящее к успешному (по смыслу примера) решению.

В результате разбора входного файла с РШТ Т3 первоначально создается *lp*-файл, который затем модифицируется на каждом шаге решения. Начальное и конечное состояния *lp*-файла представлены на рисунке 2.

Результирующий файл results.txt будет следующим:

```
clingo version 5.5.2
Reading from ast_asp.lp
Solving...
Answer: 1
res(c) res(d) res(f) res(passed)
SATISFIABLE

Models       : 1
Calls        : 1
Time         : 0.002s (Solving: 0.00s
               1st Model: 0.00s Unsat: 0.00s)
CPU Time     : 0.000s
```

Результирующий файл, помимо конечного множества убеждений, содержит еще и дополнительную информацию от clingo о времени решения. К сожалению, информацию о времени в нем нельзя считать релевантной, так как представлен результат последнего запуска clingo, а не агрегированные результаты всего множества запусков, предполагаемых в рамках решения.

Заключение

В работе была предложена архитектура решателя РШТ логики высказываний. Описан процесс разработки прототипа решателя, соответствующего предложенной архитектуре.

Для разработки решателя потребовалось реализовать анализатор, транслятор, переводящий РШТ в логическую программу ASP и перенаправляющий ее для получения множества убеждений в систему вывода clingo, а также модификатор РШТ, способный отфильтровывать правила, которые гарантированно не будут задействованы на текущем шаге прогона модели.

Рассмотрен пример, демонстрирующий поэтапный (пошаговый) процесс решения РШТ.

Предложенные программные средства зарегистрированы в реестре программ для ЭВМ РФ, свидетельство № 2021681551, и являются первой и уникальной реализацией системы вывода РШТ.

В дальнейшем планируется расширить возможности решателя введением средств для работы с логикой предикатов первого порядка.

Работа выполнена при финансовой поддержке проектов РФФИ № 20-57-00015 Бел а, 20-07-00498.

Литература

1. Панов А.И. Целеполагание и синтез плана поведения когнитивным агентом // Искусственный интеллект и принятие решений. 2018. № 2. С. 21–35. DOI: 10.14357/20718594180202.

2. Furber R., Mardare R., Mio M. Probabilistic logics based on Riesz spaces. LMCS, 2020, vol. 16, iss. 1. DOI: 10.23638/LMCS-16(1:6)2020. URL: https://lmcs.episciences.org/6054 (дата обращения: 24.02.2022).

3. Aldini A., Curzi G., Graziani P., Tagliaferri M. Trust evidence logic. Proc. ECSQARU, Lecture Notes in Computer Science, 2021, vol. 12897, pp. 575–589. DOI: 10.1007/978-3-030-86772-0_41.

4. Mitani Y., Kobayashi N., Tsukada T. A probabilistic higher-order fixpoint logic. Proc. V Int. Conf. FSCD, 2020, vol. 167, art. 19, pp. 19:1–19:22. DOI: 10.4230/LIPIcs.FSCD.2020.19.

5. Rybakov V. Temporal multi-valued logic with lost worlds in the past // Сибирские электронные математические изв. 2018. Т. 15. С. 436–449.

6. Гнатенко А.Р., Захаров В.А. О выразительных возможностях некоторых расширений линейной темпоральной логики // Моделирование и анализ информационных систем. 2018. Т. 25. № 5. С. 506–524. DOI: 10.18255/1818-1015-2018-5-506-524.

7. Elgot-Drapkin J., Perlis D. Reasoning situated in time I: Basic concepts. J. of Experimental and Theoretical Artificial Intelligence, 1990, vol. 2, pp. 75–98. DOI: 10.1080/09528139008953715.

8. Fominykh I., Eremeev A., Alekseev N., Gulyakina N. Decision making and control of a cognitive agent's knowledge process under time constraints. In: Communications in Computer and Information Science, 2020, pp. 212–221. DOI: 10.1007/978-3-030-60447-9_13.

9. Fominykh I., Vinkov M. Paraconsistency of argumentation semantics for stepping theories of active logic. Proc. I Int. Sci. Conf. IITI'16, 2016, vol. 1, pp. 171–180. DOI: 10.1007/978-3-319-33609-1_15.

10. Vinkov M., Fominykh I. Stepping theories of active logic with two kinds of negation. Advances in Electrical and Electronic Engineering, 2017, vol. 15, no. 1, pp. 84–92. DOI: 10.15598/AEEE.V15I1.1990.

11. Fominykh I., Vinkov M. Step theories of active logic and extended logical programs. In: Advances in Intelligent Systems and Computing, pp. 192–201. DOI: 10.1007/978-3-319-68321-8_20.

12. Gelfond M., Lifschitz V. The stable model semantics for logic programming. In: ICLP/SLP, 1988, pp. 1070–1080.

13. Gebser M., Kaminski R., Kaufmann B., Schaub T. Multi-shot ASP solving with clingo. Theory and Practice of Logic Programming, 2019, vol. 19, no. 1, pp. 27–82. DOI: 10.1017/S1471068418000054.

Software & Systems
DOI: 10.15827/0236-235X.138.145-152

Received 12.04.22, Revised 22.04.22
2022, vol. 35, no. 2, pp. 145–152

Development of a solver prototype for extended step theories of propositional logic

I.B. Fominykh [1], *Dr.Sc. (Engineering), Professor, fominykhIB@appmat.ru*
N.P. Alekseev [1], *Senior Lecturer, AlekseevNP@mpei.ru*
N.A. Gulyakina [2], *Ph.D. (Physics and Mathematics), Associate Professor, Head of Laboratory, guliakina@bsuir.by*
K.S. Kravchenko [1], *Graduate Student, KravchenkoKS@mpei.ru*
M.V. Fomina [1], *Ph.D. (Engineering), Associate Professor, FominaMV@mpei.ru*

[1] *National Research University "Moscow Power Engineering Institute", Moscow, 111250, Russian Federation*
[2] *Belarusian State University of Informatics and Radioelectronics (BSUIR), Minsk, 220013, Belarus*

Abstract. Nowadays, there are active researches on the possibilities of using non-classical logics in modeling the cognitive agent's reasoning.

The paper considers the problem of developing and implementing a prototype of an Extended Step Theory solver (EST) in the case when decisions on managing a complex technical object are made under strict time constraints. The authors consider a logical system based on using step theories with two types of negation, such systems are called EST. The use of two types of negation allows deducing both unbiased facts and belief facts, which is important when modeling human reasoning.

The paper focuses on the issue of organizing the inference procedure based on using non-classical logics in modeling the reasoning of a cognitive agent.

There are the main stages of the development of the EST prototype using the propositional logic literals given. There are also descriptions for each solver component, its functions, tasks, input and output data. The authors is justify the choice of the clingo output system supporting the formation of extended logic programs Answer Set Programming (ASP) as a tool for implementing the solver.

The paper gives the algorithms of translating the EST into a logical program corresponding to the ASP syntax. When organizing logical inference, the authors used the algorithm of cyclic processing of EST belief sets in the clingo environment. The main stages of this algorithm are considered by an example that analyzes the solver's operation stages and the presents the results in the clingo syntax. An example of the solver's work

demonstrates the main EST features in hard real-time problems, such as the rejection of logical omniscience, self-knowledge and temporal sensitivity.

It is planned further to consider the applicability of the created solver to a more complex formal system – the logic of first-order predicates.

Keywords: advanced stepping theory, solver, active logic, time constraints, logic programming.

Acknowledgements. *This work was supported by RFBR projects no. 20-57-00015 Bel a, no. 20-07-00498.*

References

1. Panov A.I. Goal setting and behavior planning for cognitive agent. *Artificial Intelligence and Decision Making*, 2018, no. 2, pp. 21–35. DOI: 10.14357/20718594180202 (in Russ.).
2. Furber R., Mardare R., Mio M. Probabilistic logics based on Riesz spaces. *LMCS*, 2020, vol. 16, iss. 1. DOI: 10.23638/LMCS-16(1:6)2020. Available at: https://lmcs.episciences.org/6054 (accessed February 24, 2022).
3. Aldini A., Curzi G., Graziani P., Tagliaferri M. Trust evidence logic. *Proc. ECSQARU, Lecture Notes in Computer Science*, 2021, vol. 12897, pp. 575–589. DOI: 10.1007/978-3-030-86772-0_41.
4. Mitani Y., Kobayashi N., Tsukada T. A probabilistic higher-order fixpoint logic. *Proc. V Int. Conf. FSCD*, 2020, vol. 167, art. 19, pp. 19:1–19:22. DOI: 10.4230/LIPIcs.FSCD.2020.19.
5. Rybakov V. Temporal multi-valued logic with lost worlds in the past. *Siberian Electronic Mathematical Reports*, 2018, vol. 15, pp. 436–449.
6. Gnatenko A.R., Zakharov V.A. On the expressive power of some extensions of linear temporal logic. *Modeling and Analysis of Information Systems*, 2018, vol. 25, no. 5, pp. 506–524. DOI: 10.18255/1818-1015-2018-5-506-524 (in Russ.).
7. Elgot-Drapkin J., Perlis D. Reasoning situated in time I: Basic concepts. *J. of Experimental and Theoretical Artificial Intelligence*, 1990, vol. 2, pp. 75–98. DOI: 10.1080/09528139008953715.
8. Fominykh I., Eremeev A., Alekseev N., Gulyakina N. Decision making and control of a cognitive agent's knowledge process under time constraints. In: *Communications in Computer and Information Science*, 2020, pp. 212–221. DOI: 10.1007/978-3-030-60447-9_13.
9. Fominykh I., Vinkov M. Paraconsistency of argumentation semantics for stepping theories of active logic. *Proc. I Int. Sci. Conf. IITI'16*, 2016, vol. 1, pp. 171–180. DOI: 10.1007/978-3-319-33609-1_15.
10. Vinkov M., Fominykh I. Stepping theories of active logic with two kinds of negation. *Advances in Electrical and Electronic Engineering*, 2017, vol. 15, no. 1, pp. 84–92. DOI: 10.15598/AEEE.V15I1.1990.
11. Fominykh I., Vinkov M. Step theories of active logic and extended logical programs. In: *Advances in Intelligent Systems and Computing*, pp. 192–201. DOI: 10.1007/978-3-319-68321-8_20.
12. Gelfond M., Lifschitz V. The stable model semantics for logic programming. In: *ICLP/SLP*, 1988, pp. 1070–1080.
13. Gebser M., Kaminski R., Kaufmann B., Schaub T. Multi-shot ASP solving with clingo. *Theory and Practice of Logic Programming*, 2019, vol. 19, no. 1, pp. 27–82. DOI: 10.1017/S1471068418000054.

Для цитирования

Фоминых И.Б., Алексеев Н.П., Гулякина Н.А., Кравченко К.С., Фомина М.В. Разработка прототипа решателя для расширенных шаговых теорий логики высказываний // Программные продукты и системы. 2022. Т. 35. № 2. С. 145–152. DOI: 10.15827/0236-235X.138.145-152.

For citation

Fominykh I.B., Alekseev N.P., Gulyakina N.A., Kravchenko K.S., Fomina M.V. Development of a prototype solver for extended step theories of propositional logic. *Software & Systems*, 2022, vol. 35, no. 2, pp. 145–152 (in Russ.). DOI: 10.15827/0236-235X.138.145-152.