© А.Г. ИВАШКО, А.В. ГРИГОРЬЕВ, М.В. ГРИГОРЬЕВ
ivashco@mail.ru, 107th@mail.ru, m_grigoriev@mail.ru

УДК 510.5:510.22:519-6

*МОДИФИКАЦИЯ ТАБЛИЧНОГО АЛГОРИТМА НА ОСНОВЕ ПРОВЕРКИ НЕПЕРЕСЕКАЕМОСТИ СЛОЖНЫХ КОНЦЕПТОВ**

АННОТАЦИЯ. Статья посвящена исследованию необходимости изменения табличного алгоритма с целью уменьшения времени работы для обеспечения возможности его использования при проверке баз знаний больших объемов.

SUMMARY. The article considers researching the necessity of tableau algorithm changing with aim to reduce algorithm work time, for providing an opportunity of its using for checking large knowledge bases.

КЛЮЧЕВЫЕ СЛОВА. Имитационное моделирование, дескрипционная логика, концепт, роль, табличный алгоритм.

KEYWORDS. Simulation modeling, description logic, concept, role, tableau algorithm.

Одним из широко используемых инструментов представления знаний является язык дескрипционной логики ALC [1-3]. Представление состоит из терминологического описания концептов и их ролевых отношений (*TBox*) и описания фактов, имеющих место в предметной области (*ABox*). Применение дескрипционной логики дает возможность осуществлять вывод знаний, неявно содержащихся в описании терминологии в виде GCI-отношений [1], а также осуществлять проверку непротиворечивости логики описания предметной области с помощью табличного алгоритма. Табличный алгоритм является имитационным алгоритмом построения непротиворечивой интерпретации предметной области на основе терминологического описания концептов и их ролей, представленных в *TBox*. Под интерпретацией понимается набор объектов предметной области (могут быть вымышленные объекты), причем все концепты, описанные в *TBox*, должны быть представлены хотя бы одним объектом [4-5]. Связь между объектами определяется ролевыми отношениями концептов. Интерпретация является непротиворечивой, если не существует ни одного объекта, который принадлежит концепту и его дополнению. Представление знаний является корректным, если существует хотя бы одна непротиворечивая интерпретация предметной области. Табличный алгоритм имеет экспоненциальную сложность, так как при отнесении объекта к концепту, являющимся дизъюнкцией других концептов, он должен быть отнесен к одному из операндов, соответственно будет порождено 2^n таблиц, где n — количество дизъюнктивных концептов. Существуют оптимизационные техники [6], которые сокращают количество выполняемых операций, однако табличный алгоритм все же медленно работает при проверке баз знаний больших объемов.

* Работа выполнена по проекту «Методология построения управляемой архитектуры предприятия на основе онтологического представления информационных систем» в соответствии с заданием Министерства образования и науки Российской Федерации на 2012 год.

Целью данной работы является модификация табличного алгоритма, которая позволит ускорить построение непротиворечивой интерпретации для сложных концептов, состоящих из объединения простых.

Каждый сложный концепт в дескрипционной логике может быть представлен в виде дерева, где каждое поддерево будет представлять собой отдельный концепт, причем каждый лист будет представлять простой концепт или его дополнение, а каждый внутренний узел дерева будет представлять конструктор:

- конъюнкция концептов $\prod_i x_i$ будет представлена деревом с корнем, имеющим метку ⊓ и потомками, каждый из которых будет представлять концепт x_i;

- дизъюнкция концептов $\bigsqcup_i x_i$ будет представлена деревом с корнем, имеющим метку ⊔ и потомками, каждый из которых будет представлять концепт x_i;
- концепт $\exists r.C$, будет представлен деревом с корнем, имеющим метку $\exists r$ (где r — роль) и единственным потомком, представляющим концепт C;
- концепт $\forall r.C$, будет представлен деревом с корнем, имеющим метку $\forall r$ (где r — роль) и единственным потомком, представляющим концепт C;
- простой концепт C ($\neg C$) будет представлен единственным узлом с меткой C ($\neg C$).

Пример сложного концепта, включающий перечисленные выше конструкции, представлен на рисунке 1. Для идентификации узлов деревьев используется их нумерация.

Модификация табличного алгоритма заключается в том, чтобы определять может ли существовать непротиворечивая интерпретация еще до полного раскрытия всех правил. Покажем что данное утверждение корректно.

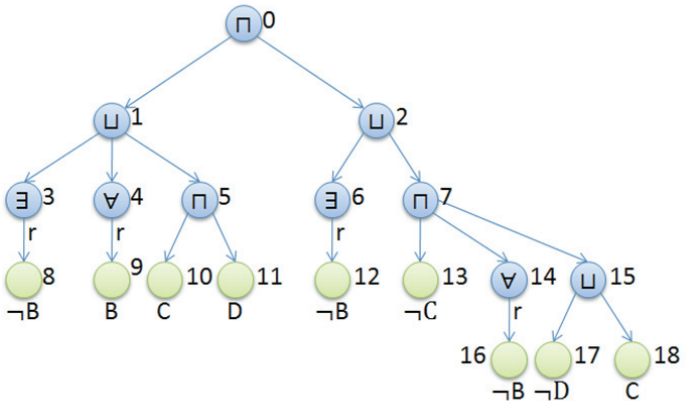

Рис. 1. Дерево, представляющее сложный концепт:
(∃r.¬B⊔∀r.B⊔C⊓D)⊓(∃r.¬B⊔⊓(¬C⊓∀r.¬B⊓(¬D⊔C)))

Теорема 1. Если в непротиворечивой интерпретации I существует объект x, который принадлежит множеству концептов $\{C_1,C_2,...,C_n\}$, тогда, если существует такой концепт D, что $\exists C_i \sqsubseteq \neg D$, то при добавлении этого концепта к множеству $\{C_1,C_2,...,C_nD\}$, то $\exists C \in \{C_1,C_2,...,C_n,D\} : x \in C$, $x \in \Delta^i / C^i$.

Доказательство.

Предположим, что $\forall i : \exists x : x \in C_i^I$, так как $x \in D^I$, то $x \notin \Delta^I \ / \ D^I$, а так как $\exists i : C_i^I \sqsubseteq \neg D$, то $x \notin C_i^I$ при том, что $x \in C_i^I$, а значит интерпретация является противоречивой. ■

Пусть $R(x)$ является концептом, представленным поддеревом с вершиной x. Обозначим всех прямых потомков вершины x, как $P(x)$, соответственно i-го прямого потомка обозначим как $P_i(x)$. Тогда для определения того, пересекаются ли концепты $R(x)$ и концепт $R(y)$ может быть применен следующий алгоритм.

Модифицированный алгоритм построения интерпретации $DJ(x,y)$:

1. if $R(x) \in D(R(y))$ or $R(y) \in D(R(xy))$ then return true;
2. if x имеет метку $\sqcup$ and y имеет метку $\sqcap$ then
3. if $\forall p_1 \in P(x) \exists p_2 \in P(y) : DJ(R(p_1), R(p_2))$ then
4. $R(y) \to D(R(x))$
5. return true;
6. if x имеет метку $\sqcup$ and y имеет метку $\sqcup$ then
7. if $\forall p_1 \in P(x)$, $\forall p_2 \in P(y)$, $DJ(R(p_1), R(p_1))$ then
8. $R(y) \to D(R(x))$
9. return true;
10. if x имеет метку $\sqcup$ and y представляет простой концепт *or* дополнение простого концепта *or* имеет метку $\exists r$, $\forall r$ then
11. if $\forall p \in P(x)$: $DJ(R(p), R(y))$ then $R(y) \to D(R(x))$, return true;
12. if x имеет метку $\sqcap$ and y имеет метку $\sqcap$ then
13. if $\exists p_1 \in P(x) \exists p_2 \in P(y) : DJ(R(p_1), R(p_2))$ then
14. $R(y) \to D(R(x))$
15. return true;
16. if x имеет метку $\sqcap$ and y имеет метку $\sqcup$ then
17. if $\forall p_2 \in P(y) \exists p_1 \in P(x) : DJ(R(p_1), R(p_2))$ then
18. $R(y) \to D(R(x))$
19. return true;
20. if x имеет метку $\sqcap$ and if y представляет простой концепт *or* дополнение простого концепта *or* имеет метку $\exists r$, $\forall r$ then
21. if $\exists p \in P(x) : DJ(R(p), R(y))$ then
22. $R(y) \to D(R(x))$
23. return true;
24. if x имеет метку $\exists r$ and y имеет метку $\forall r$ and роли вершин совпадают then
25. if $DJ(P_1(p), P_1(y))$ then
26. $R(y) \to D(R(x))$
27. return true;
28. if x имеет метку $\forall r$ and y имеет метку $\exists r$ and роли вершин совпадают then

29. if $DJ(P_1(x), P_1(y))$ then
30. $R(y) \rightarrow D(R(x))$
31. return true;
32. if x представляет простой концепт and y представляет простой концепт then
33. if $R(x) \sqsubseteq \neg R(y)$ then
34. $R(y) \rightarrow D(R(x))$
35. return true;
36. return false;

Check(C,E)

1. Перевести концепты C и E в древовидную структуру;
2. $x \leftarrow$ корень дерева, представляющего концепт C;
3. $y \leftarrow$ корень дерева, представляющего концепт E;
4. if $DJ(x,y)$ then
5. return C не пересекается с E;

Покажем, что данный алгоритм выполняется корректно.

Лемма 1. Если концепт C является дизъюнкцией концептов $c_1 \sqcup c_2 \sqcup ... \sqcup c_k$ и концепт D является дизъюнкцией концептов $d_1 \sqcup d_2 \sqcup ... \sqcup d_h$ и $\forall i,j : c_1 \sqsubseteq \neg d_j$ при $i = i..k$, $j = 1..h$, то $C \sqsubseteq \neg D$ в любой непротиворечивой интерпретации I.

Доказательство.

Предположим, что в некоторой интерпретации J $\exists x : x \in C^J$ и $x \in D^J$, тогда $\exists i : x \in c_i^J$ при i=1..k и $\exists j : x \in d_i^J$ при j=1..h, то $x \notin \Delta^I \setminus d_j^I$, но так как по условию леммы $\forall i,j : c_i \sqsubseteq \neg c_j$, то $x \notin c_i^J$, а значит интерпретация J является противоречивой, значит $C \sqsubseteq \neg D$ в любой непротиворечивой интерпретации. ■

Лемма 2. Если концепт C является дизъюнкцией концептов $c_1 \sqcup c_2 \sqcup ... \sqcup c_k$ и концепт D является конъюнкцией концептов $d_1 \sqcap d_2 \sqcap ... \sqcap d_h$ и $\forall i \exists j : c_i \sqsubseteq \neg d_j$ при $i = i..k$, $j = 1..h$, то $C \sqsubseteq \neg D$ в любой непротиворечивой интерпретации I.

Доказательство.

Предположим, что в некоторой интерпретации J $\exists x : x \in C^J$ и $x \in D^J$, тогда $\exists i : x \in c_i^J$ при i=1..k и $\forall j : x \in d_j^J$ при j=1..h, то $x \notin \Delta^I \setminus d_j^I$, но так как по условию леммы $\forall i \exists j : c_i \sqsubseteq \neg d_j$, то $x \notin c_i^J$, а значит интерпретация J является противоречивой, значит $C \sqsubseteq \neg D$ в любой непротиворечивой интерпретации. ■

Лемма 3. Если концепт C является конъюнкцией концептов $c_1 \sqcap c_2 \sqcap ... \sqcap c_k$ и концепт D является конъюнкцией концептов $d_1 \sqcap d_2 \sqcap ... \sqcap d_h$ и $\exists i \exists j : c_i \sqsubseteq \neg d_j$ при $i = i..k$, $j = 1..h$, то $C \sqsubseteq \neg D$ в любой непротиворечивой интерпретации I.

Доказательство.

Предположим, что в некоторой интерпретации J $\exists x : x \in C^J$ и $x \in D^J$, тогда $\forall i : x \in c_i^J$ при i=1..k и $\forall j : x \in d_j^J$ при j=1..h, то $x \notin \Delta^I \setminus d_j^I$, но так как по условию леммы $\exists i \exists j : c_i \sqsubseteq \neg d_j$, то $x \notin c_j^J$, а значит интерпретация J является противоречивой, значит $C \sqsubseteq \neg D$ в любой непротиворечивой интерпретации. ■

Лемма 4. Если $C \sqsubseteq \neg D$, то $x \in C^I$ и $x \notin D^I$ в любой непротиворечивой интерпретации I.

Доказательство.

Рассмотрим произвольную интерпретацию J. Предположим, что существует такой объект $x \in C^I$ и $x \in D^I$, тогда $x \notin \Delta^I \setminus D^I$, значит $x \in \notin C^I$, но т.к. $x \in C^I$, то рассматриваемая интерпретация является противоречивой. ■

Лемма 5. Если концепт C является образуется с применением конструктора $\exists R.F$ и концепт D образуется с применением конструктора $\forall P.E$ и $R=P$ и $E \sqsubseteq \neg F$, тогда $C \sqsubseteq \neg D$.

Доказательство.

Предположим, что в некоторой интерпретации J $\exists x : x \in C^J$ и $x \in D^J$, тогда объект x имеет R-последователя y, такого, что $y \in E^J$, $y \in F^J$, однако по лемме 4, интерпретация J является противоречивой. ■

Теорема 2.

Если при выполнении алгоритма 1, значение $Check(C,E)$ истинно, тогда $E^I \subseteq \Delta^I / C^I$ в любой интерпретации I.

Доказательство.

Докажем по индукции от высоты h дерева, представляющего концепт C и высоты дерева t, представляющего концепт E:

При h=1, t=1, C и E являются простыми концептами, тогда функция DJ вернет истинное значение только при $E \sqsubseteq \neg C$ и по лемме 4 $E^I \subseteq \Delta^I / C^I$;

1. Предположим, что функция $Check(C,E)$ работает корректно при всех $h=h_1$, и $t=t_1$. Покажем, что функция работает корректно при $h=h_1+1$ и $t=t_1+1$, тогда концепт C имеет высоту h_1+1, и концепт E имеет высоту t_1+1.

Рассмотрим 4 случая:

а) если концепт C является конъюнкцией других концептов, и концепт E является дизъюнкцией, или концепт C является дизъюнкцией других концептов, и концепт E является конъюнкцией концептов, тогда по лемме 2 выполняется $E^I \subseteq \Delta^I / C^I$;

б) если концепт C является конъюнкцией других концептов, и концепт E является конъюнкцией концептов, тогда по лемме 3 выполняется $E^I \subseteq \Delta^I / C^I$;

в) если концепт C является дизъюнкцией других концептов, и концепт E является дизъюнкцией концептов, тогда по лемме 1 выполняется $E^I \subseteq \Delta^I / C^I$;

г) если концепт C образуется с применением конструктора $\exists R.F$, и концепт E образуется с применением конструктора $\forall P.D$ и $R=P$, тогда по лемме 5 выполняется $E^I \subseteq \Delta^I / C^I$;

2. Предположим, что функция $Check(C,E)$ работает корректно при всех $h = h_1$ и $t = t_1$. Покажем что функция работает корректно при $h=h_1$ и $t=t_1+1$, тогда концепт C имеет высоту h_1, и концепт E имеет высоту t_1+1.

Рассмотрим 2 случая:

а) если концепт C образуется с применением конструктора $\exists R.F$ или конструктора $\forall R.F$ и концепт E является дизъюнкцией, тогда по лемме 1 при том, что концепт C является дизъюнкцией одного дизъюнкта, выполняется $E^I \subseteq \Delta^I / C^I$;

б) если концепт C образуется с применением конструктора $\exists R.F$ или конструктора $\forall R.F$, и концепт E является конъюнкцией, тогда по лемме 2 при том, что концепт C является дизъюнкцией одного дизъюнкта, выполняется $E^I \subseteq \Delta^I / C^I$. ■

Рассмотрим пример работы табличного алгоритма с применением данной модификации.

Пусть множество аксиом *TBox* состоит из следующих аксиом:

1. Студент ⊑ Участник ⊓ ∃выполнять.Контрольная работа;
2. Участник ⊔ ∃выполнять.Контрольная работа ⊑ ¬Преподаватель ⊓ Участник;
3. Студент ⊑ ¬Контрольная работа;
4. Преподаватель ⊑ ¬Контрольная работа;
5. Преподаватель ⊑ Участник;

В начале работы табличного алгоритма каждая аксиома будет переведена в вид сложного концепта:

1. ¬Студент ⊔ Участник ⊓ ∃выполнять.Контрольная работа;
2. ¬Участник ⊔ ∀выполнять.¬Контрольная работа ⊔ ¬Преподаватель ⊓ Участник;
3. ¬Студент ⊔ ¬Контрольная работа;
4. ¬Преподаватель ⊔ ¬Контрольная работа;
5. ¬Преподаватель ⊔ Участник;

После чего будет создан объект интерпретации *x,* отнесенный к произвольному простому концепту (в данном случае к концепту Студент), и ко всем переведенным аксиомам. Затем каждый из сложных концептов будет раскрыт в соответствии с правилами табличного алгоритма [1], [3]. При раскрытии первого концепта, не может быть выбран концепт ¬Студент, так как в этом случае объект *x* будет отнесен в концептам ¬Студент и Студент, что приведет к противоречивой интерпретации, соответственно будет выбран концепт Участник ⊓ ∃выполнять.Контрольная работа. При раскрытии второго концепта, необходимо выбрать один из операндов дизъюнкции, представленного в виде дерева на рис. 2, и проверить на непересекаемость с уже добавленными концептами, представленными на рисунке 3.

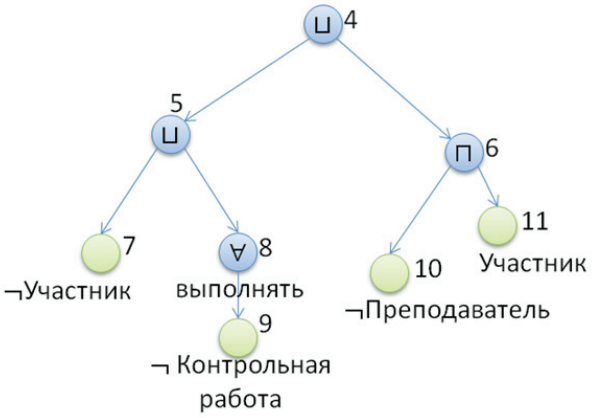

Рис. 2. Раскрываемый концепт

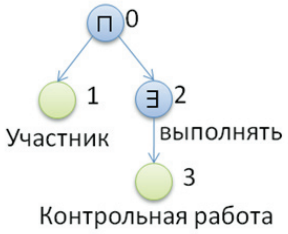

Рис. 3. Добавленный концепт

При добавлении левого поддерева дерева, представленного корнем в вершине 4, функция $Check(R(0),R(5))$ сообщит о том, что данные концепты не пересекаются, так как $R(5) \in D(R(0))$, потому что $R(7) \in D(R(1))$, так как концепты «Участник» и «¬Участник» являются непересекающимися и $R(8) \in D(R(2))$, так как обе вершины имеют роль «выполнять» и $R(9) \in D(R(3))$, так как концепты «Контрольная работа» и «¬Контрольная работа» являются непересекающимися.

Непротиворечивая интерпретация построенная табличным алгоритмом, примет вид, представленный на рисунке 4.

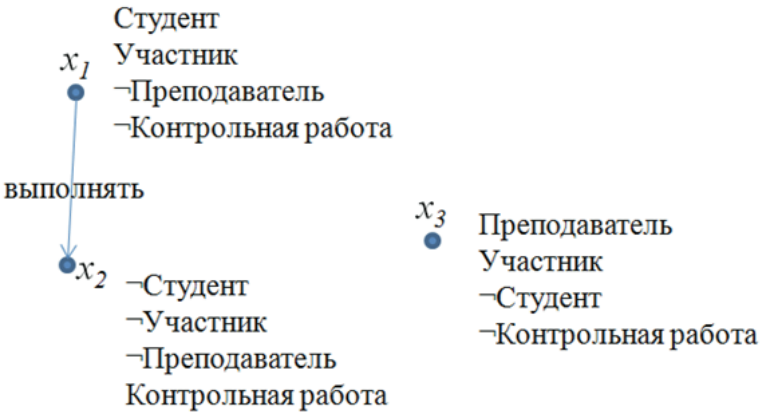

Рис. 4. Интерпретация построенная табличным алгоритмом

Выводы

1. Разработана модификация табличного алгоритма, которая позволит уменьшить количество выполняемых операций за счет сокращения перебора заведомо неправильных вариантов выбора при раскрытии объединения концептов. Это в свою очередь дает возможность использовать его при проверке баз знаний больших объемов, как, например, антологическое представление архитектуры информационных систем.

2. Доказана корректность работы модифицированного алгоритма для любых баз знаний, описанных на языке дескрипционной логики *ALC*.

СПИСОК ЛИТЕРАТУРЫ

1. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., and Patel-Schneider, P.F., editors. The Description Logic Handbook: Theory, Implementation and Applications. CUP, 2003. P. 601.

2. Глухих И.Н. Интеллектуальные системы проблемного обучения: концепция, управление и модели знаний // Вестник ТюмГУ. 2002. №3. С. 21-27.

3. Sebastian Rudolph. Foundations of Description Logics // Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, Peter F. Patel-Schneider, Reasoning Web. Semantic Technologies for the Web of Data — 7th International Summer School 2011, Springer, LNCS, Vol.6848, 2011, P. 76-136.

4. Ивашко А.Г., Григорьев М.В. Объектно-ориентированный язык ограничений для верификации процесса командной разработки программного обеспечения // Вестник ТюмГУ. 2008. №6. С. 152-158.

5. Ивашко А.Г., Коломиец И.И. Возможность применения аппарата сетей Петри для валидации анализа бизнес-процессов // Вестник ТюмГУ. 2008. №6. С. 159-165.

6. Tsarkov, D. and Horrocks, I. FaCT++ Description Logic Reasoner: System Description // Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006), V. 4130 of Lecture Notes in Artificial Intelligence, Springer, 2006, P. 292-297.