УДК 519.68

ОТ МАТЕМАТИЧЕСКОЙ ЛОГИКИ К ЯЗЫКАМ ПРОГРАММИРОВАНИЯ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

В. Куклин

Харьковский национальный университет имени В.Н. Каразина, пл. Свободы, 4, г. Харьков, 61022, Украина
kuklinvm1@gmail.com

Рецензент: Александр Потий, д.т.н., проф., Харьковский национальный университет имени В. Н. Каразина, пл. Свободы, 4, г. Харьков, 61022, Украина.
potav@ua.fm

Поступила в январе 2017

Аннотация. *Рассмотрен процесс становления теории экспертных систем на примере формирования языка программирования искусственного интеллекта ПРОЛОГ. Показан сложный путь осознания проблем искусственного интеллекта и мотивы, которые привели к появлению экспертных систем, построенных на основе математической логики. Обсуждаются основные идеи и процедуры, которые привели к построению сначала отдела математической логики - теории предикатов, представлению процедур этой теории на гиперграфах и затем к созданию языка ПРОЛОГ. Отмечается прогресс в развитии интеллектуальных систем и проблемы, которые стоят перед исследователями.*

Ключевые слова*: теория предикатов, гиперграфы И/ИЛИ, язык программирования ПРОЛОГ.*

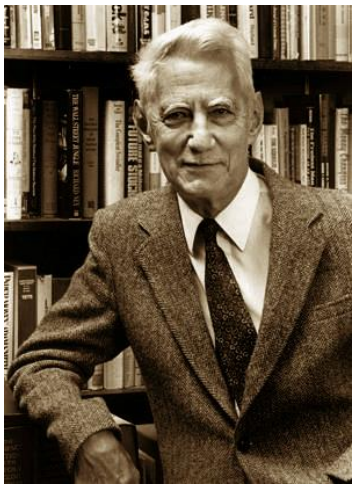
1 Введение

Развитие вычислительной техники стимулировало развитие программного обеспечения, особенно языков программирования высокого уровня. Вообще языки программирования обычно создавали по принципу удобства решения какого-либо класса задач. При этом большинство из них к математике имели весьма отдаленное отношение. Однако для языков искусственного интеллекта ситуация сложилась иначе. Но все по порядку.

Аллен Ньюэлл
(Allen Newell)

Работал в RAND и Университете Карнеги-Меллон. Разработал программмы Logical Theorist (*доказавшую законы из книги Рассела и Уайтхеда «Principia Mathematica»*) и «General Problem Solver» для игры в шахматы. Один из создателей языка IPL.

Клод Э́лвуд Ше́ннон
(Claude Elwood Shannon)



Докторская диссертация в Массачусетском технологическом институте, работа в компании Белл (Bell Labs) и Мичиганском университете. Основатель теории информации и теории автоматов.
Впервые использовал понятие «бит».

Фридрих Людвиг Готлоб Фреге
(Friedrich Ludwig Gottlob Frege)

Вклад Фреге в логику сравнивают с вкладом Аристотеля и К. Геделя. В работе *Begriffsschrift* (Исчисление понятий) Фреге (1879) создал логику предикатов, ввел кванторы и многое другое, что способствовало появлению книги «Principia Mathematica» и теоремы о неполноте Геделя.

© Куклин В., 2017

Итак, после позднего увлечения шахматами Алленом Ньюэлом, создавшим действующую программу (1954) [1,2] на основе методов Клода Шеннона [3], массово потянулись энтузиасты-последователи. Но о математической логике, которая была давно создана благодаря усилиям Фридриха Готлоба Фреге [4] и его современников, пока никто не вспоминал, что, впрочем, не удивительно. Игры показались интереснее. Однако, пригодный метод для решения задачи создания шахматных программ практически без применения формализма предложил Алан Тьюринг [5]. Не надеясь на математику он вместе с Алонзо Чёрчем считал, что все известные им программы настолько склонны увлекаться собственными действия, что остановить их можно лишь насильственно *(тезис Черча-Тьюринга)*. Потому в программы так часто вводятся операторы и процедуры, сдерживающие прыть машин, которые самозабвенно «роются» в базах данных, в то время, когда пользователю, иной раз, некогда ждать. В результате усилий сотрудников корпорации RAND Джона Шоу и Герберта Саймона, поддерживаемых де Гротом и его коллегами психологами был создан язык ИПЛ (IPL, 1956), предшественник языка ЛИСП *(появился позднее в результате усилий Джона Маккарти (1960))*. Язык ЛИСП - язык обработки списков, был построен на системе лямбда-вычислений А. Черча, как и другие функциональные языки этого семейства, разработанные последователями [6-8].

Алан Тьюринг
(Alan Mathison Turing)

А. Тьюринг внес существенный вклад в основания информатики, разработал теорию и метод взлома немецкого шифратора во время 2-й мировой войны.
Работал в Национальной физической лаборатории и университете Манчестера. Создатель проекта первого компьютера с памятью. Предложил тест своего имени для оценки интеллекта (1950).

Герберт АлександерСаймон
(Herbert A. Simon)

Разработчик теории организации и управления. Один из создателей современной теории управленческих решений *(теория ограниченной рациональности)*.
Профессор компьютерных наук и психологии в Университете Карнеги-Меллона. Нобелевская премия по экономике (1978).

Жак Эрбран
(Jacques Herbrand)

Основные труды в математической логике, теории полей классов, ввел рекурсивные функции, разработал математический метод резолюции.
Теорема Эрбрана о дедукции является результатом диссертации по теории доказательств.

Вообще интересные люди – эти последователи: обыкновенно получив в свое распоряжение новый метод, человек сразу ищет, где бы его применить и как бы его модифицировать, не только для удобства решения нужной проблемы, но и для создания своего собственного творения. При этом последний мотив настолько распространен, что многие научные журналы переполнены вариантами решения задач, которые при ближайшем рассмотрении мало отличаются от самой первой, где собственно и был впервые применен этот тиражируемый метод *(поделки, хотя возможно и небесполезные)*.

Использование формальной логики для создания логического метода «опровержения на основе резолюции» Джоном Робертсоном (1965) оказалось революционным [9, 10].

На основе этого метода, хотя и с целым рядом отличий, был создан язык ПРОЛОГ Алайном Колмраузером, (1971), использующий логику предикатов первого порядка [11]. Кстати, математический метод резолюции был ранее применен юным математиком Жаком Эрбраном [12]. То есть, к началу 70-х годов были созданы языки программирования искусственного интеллекта ЛИСП, ПРОЛОГ, PLANNER, QA4, MACSYMA, REDUCE, TSM и их модификации, ориентированные на выполнение разных задач.

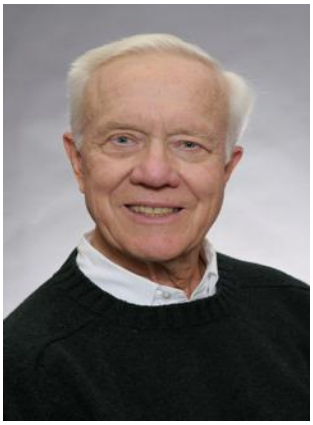
Джон Алан Робинсон
(John Alan Robinson)

Внес определяющий вклад в развитие логического программирования, докторская диссертация в Принстонском университете.
Работал в концерне DuPoint и в университете Райса, где сформулировал практический метод резолюции в логике.

А. Колмерауер
(Alain Colmerauer)

Защитил диссертацию в университете Гренобля.
Работал профессором в университете Монреаля *(где создал Q-system)*, затем в университете Марселя.
Создатель языка ПРОЛОГ.

Нильс Нильсон
(Nils J. Nilsson)



Руководитель Центра искусственного интеллекта Стенфордского университета.
Провел анализ свойств резолюции, выпустил несколько учебников *(в т.ч. «Принципы искусственного интеллекта»)*.

В 70-е годы 20-го века многие ученые и инженеры серьезно занялись отработкой формализма, стандартизованным представлением знаний в символьной форме, понятной машинам. На этом пути стало очевидно, что устройства на базе языков ИИ (искусственного интеллекта) должны сами собирать и обрабатывать информацию, не нуждаясь при этом в традиционных алгоритмах программистов, которые только наполняли бы глобальную базу данных, а что с этими данными делать впоследствии, решала бы уже сама программа. То есть языки программирования ИИ стали кардинально отличаться от обычных языков программирования, ибо в эти языки был встроен механизм решения задач.

Но, главное – это то, что появились языки программирования искусственного интеллекта, основанные на математике. Используя эти языки программирования стало возможным не только просмотреть весь ход решения задачи, но и поправить ее решение, в случае если что-то не нравилось или шло не так.

Другими словами программисты стали видеть, что происходит со знаниями, которые они вводили в память вычислительной машины на всех этапах решения задачи. Это обстоятельство, конечно, отличалось от нейронных систем, где понять, что там «творится» в этом «черном ящике» – нейрокомпьютере, было невозможно. Так как знания в память вычислительной машины вводились на основе данных и оценок экспертов, то и системы стали называть экспертными. Ниже детальнее рассмотрим основные идеи и процедуры, которые позволили создать язык ИИ – ПРОЛОГ.

2 Математическая логика и теория предикатов

Как известно, основой логики является суждение. В математической логике – это последовательность *(важен порядок следования)* квантор, субъект, связка и предикат (лат. *praedicatum* – сказанное). Для связок и кванторов в теории предикатов, которая есть частью математической логики, приняты следующие обозначения.

Таблица 1 – Принятые обозначения

СВЯЗКИ	***КВАНТОРЫ***
$\wedge$, * коньюнкция – И $\vee$, + дизьюнкция – ИЛИ* $\Rightarrow$ импликация – ЕСЛИ...ТО ** -- отрицание - НЕ	$\exists$ *существования* ($\exists$ x - *найдется такой x)* $\forall$ *общности* *(*$\forall$ *x - для каждого x)*

*$\dot{\vee}$ - *возможна дизьюнкция в строго разделительном смысле.*
***ЕСЛИ - антецедент, посылка ТО – консеквент, заключение.*

Таким образом суждение можно записать, например, так

$$(\forall x)\,[СЛОН\,(x),\ \Rightarrow ЦВЕТ\,(x,\ СЕРЫЙ]\,. \qquad (1)$$

Если обозначить символом T истину, а символом F – ложь, то вполне очевидными будут являться утверждения представленные в Табл. 2.

Таблица 2 – Принятые обозначения

$T \vee T$ $T \vee F$ $F \vee F$	*это T* *это T* *это F*
$T \wedge TT \wedge F$ $F \wedge F$	*это T* *это F* *это F*

В табл. 3 приведены не вполне очевидные, но правильные в математической логике утверждения.

Таблица 3 – Пример утверждений

--X1 $\vee$ ***X2***	*эквивалентно*	***X1*** $\Rightarrow$ ***X2***
$-T \vee T$ $-F \vee T$ $-T \vee F$ $-F \vee F$	*это T* *это T* *это F* *это T*	$T \Rightarrow T$ $F \Rightarrow T$ $T \Rightarrow F$* $T \Rightarrow T$
* *Ложно такое утверждение: «из истинного утверждения следует ложное утверждение».*		

В общем случае, можно говорить о трех видах предложений. Это факты, обычные предложения и правила (Табл. 4).

Таблица 4 – Виды предложений

ФАКТЫ Литералы	ПРЕДЛОЖЕНИЯ	ПРАВИЛА
P, P(x)	G(y) $\wedge$ S(z) --D(y) $\vee$ Q(z)	L(y) $\Rightarrow$ H(x)

Важно отметить, что в теории предикатов все связки можно заменить только двумя. Для этого нужно договориться, что все предложения, в которых используется только дизьюнкция и отрицание, должны быть связаны, например, коньюнкцией по умолчанию (так называемая, коньюктивная форма). Т.е. если сформированы два предложения, то их можно переписать в виде одного, поставив между ними связку - коньюнкцию. Возможна и дизьюнктивная форма, когда все предложения содержат только коньюнкцию и отрицание, но при этом по умолчанию связаны дизьюнкцией. Однако это форма менее принята в теории предикатов.

Интересно, что в теории предикатов импликацию заменяют на конъюнкцию, например,

$$\boldsymbol{X1} \Rightarrow \boldsymbol{X2} \text{ эквивалентно } \boldsymbol{--X1} \vee \boldsymbol{X2} \quad (2).$$

Кроме того, существуют и другие виды преобразования предложений, которые используют все связки (представлены ниже). В теории предикатов существует множество устоявшихся формул, подобно тому, как это сделано в элементарной алгебре (Табл. 4-5).

Таблица 5 – Формулы теории предикатов

ПРАВИЛА де МОРГАНА	
$--(X1 \vee X2)$ *эквивалентно* $--X1 \wedge --X2$	$--(X1 \wedge X2)$ *эквивалентно* $--X1 \vee --X2$

Таблица 6 – Формулы теории предикатов

ПРЕОБРАЗОВАНИЕ КВАНТОРОВ ОБЩНОСТИ $\forall$	ПРЕОБРАЗОВАНИЕ КВАНТОРОВ СУЩЕСТВОВАНИЯ $\exists$
$(-- \forall x)[P(x)]$ *эквивалентно* $(\exists x)[--P(x)]$	$--(\exists x)P(x)$ *эквивалентно* $(\forall x)[--P(x)];$

Обычно кванторы переносят в начало предложения и исключают сразу кванторы общности. С кванторами существования сложнее. Для работы с ними используют так называемые сколемовские функции и константы, то есть такие функции и константы, которые наверняка есть, но мы их пока не знаем. Тогда можно заменить квантор существования этими величинами, например, в следующем виде

$$(\forall y)[(\exists x)P(x,y)] \Rightarrow (\forall y)[P(g(y),y)] \ , \quad (3)$$

$$(\exists x)P(x) \Rightarrow P(A), \quad (4)$$

Здесь A и $g(y)$ – сколемовские константа и функция. Используя эти формулы можно преобразовывать сложносочиненные предложения. А процедуры непростых переходов с естественного языка на язык предикатов хорошо представлены в работе [13,14]. Рассмотрим пример преобразования сложносочиненного предложения в три бинарных.

Таблица 7 – Пример преобразования

$(\forall x)\{P(x) \Rightarrow \{(\forall y)[P(y) \Rightarrow P(f(x,y))] \wedge \sim(\forall y)[Q(x,y) \Rightarrow P(y)]\}\}$
$\sim P(x1) \vee \sim P(y) \vee P(f(x1,y))$ $\sim P(x2) \vee Q(x2,g(x2))$ $\sim P(x3) \vee \sim P(g(x3))$

Здесь при записи отдельных предложений *(традиционно для теории предикатов)* меняют аргументы, делая их независимыми. Итак, глобальная база данных состоит из фактов *(литералов)*, предложений и правил (которые можно перевести в предложения) по умолчанию связанных коньюнкцией.

Рассмотрим, как из двух предложений создать новое. Для этого в теории предикатов существует ***процедура и ее результат – резолюция***. Эта процедура является важнейшим механизмом генерации нового знания (см. Табл. 8).

Таблица 8 – Пример резолюции

РЕЗОЛЮЦИЯ - Создание нового предложений из двух	
Резолюция двух предложений $Q \vee P$ и $--P \vee G$ Ставим между предложениями дизъюнкцию $\vee$ $Q \vee P \vee --P \vee G$ Выделяем тавтологию $Q \vee (P \vee --P) \vee G$ Убираем тавтологию* $Q \vee G$.	Эти два предложения можно представить как $Q \vee P$ и $P \Rightarrow G$ при этом P заменяется на G и получаем $Q \vee G$.
* *Всегда без ущерба можно отбросить литерал или часть предложения, если они принимают заведомо истинное значение.*	

Следующей, по порядку, но не по важности процедурой в решении логических задач является ***метод опровержения на основе резолюции***. Опровержение на основе резолюции – это доказательство истинности целевой функции методом «от противного»: Задана целевая функция - в форме предложения или правила (1). Берем ее отрицание (2). Переводим его в форму предложения (3). Применяем резолюцию, то есть получаем все новые и новые предложения…(4). Если в результате подстановок получим пустое предложение, то мы доказали, что целевая функция истинна. Достоинство этого подхода заключается в его простоте и сравнительно коротком пути достижения результата.

Пример. Сформулируем задачу: Если Жора ходит в те же самые места, куда ходит Коля, а Коля в школе, то где же Жора?

$$\text{Факт: } B(KOLIA, SCHOOL); \quad (5)$$

$$\text{Правило: } \forall(x)[B(KOLIA, x) \Rightarrow B(GORA, x)]; \quad (6)$$

$$\text{Цель (вопрос): } (\exists x)B(GORA, x)). \quad (7)$$

На основной вопрос задачи «Где Жора?» можно ответить, если существует решение, иначе говоря, если $(\exists x)B(GORA, x))$ есть следствием начальной базы знаний (аксиом), то есть набора фактов и правил.

Здесь $(\exists x)B(GORA, x))$ - целевое предложение. Его отрицание (см. табл. 6) - это $(\forall x)[\sim B(GORA, x)]$.

Резолюция двух предложений $\sim B(KOLIA, y) \vee B(GORA, y)$ и $B(KOLIA, SCHOOL)$ дает ответ $B(GORA, SCHOOL)$ при унификации $y = KOLIA$.

Унификация – это согласование аргументов двух литералов, для того, чтобы можно было применить резолюцию. Использование резолюции также рассмотрено в работе [15].

Важность унификации при доказательстве теорем впервые подчеркнули авторы работ [16, 17], причем на необходимость согласованности подстановок указывает [17]. Тем не менее, вопрос о единственности решения, оставаясь открытым, вызвал большой интерес математиков, о результатах исследований которых практики мало что знают.

3 Представление теории предикатов на гиперграфах «и/или»

Рассмотрим представление элементов теории предикатов на графах. Родоначальником теории графов считается Л. Эйлер. Но как общемировая наука теория графов начала свое распространение после появления первой монографии на эту тему Д. Конига [18]. Это еще раз подтверждает известный тезис о важности популяризации научных результатов.

Леонард Эйлер (1707-1783)
Leonhard Euler

(портрет кисти Я.Э. Хандманна)

Автор фундаментальных работ по математике, математической физике, механике, оптике, кораблестроению и теории музыки.
Академик Берлинской, Туринской, Лиссабонской и Парижской наук.

Денеш Кёниг (1884-1944)
Kőnig Dénes

Докторская степень (1907). Профессор Будапештского университета. Под влиянием работ Г. Минковского *(Hermann Minkowski)* написал первую монографию по теории графов «Теория конечных и бесконечных графов» (1936).

После появления этой монографии множество исследователей разработали различные виды графов, создали большое число приложений, однако ниже нас будут интересовать такие графы, которые можно использовать для интерпретации теории предикатов. Именно такие разработанные в работах [19] описания гиперграфов И/ИЛИ можно было применять в частности и для теории предикатов.

Использование графов в исчислении предикатов часто позволяет упростить, сократить процедуры и сделать их наглядными. Пример построения графа И/ИЛИ . Важно отметить, что здесь встречается конъюнкция и дизъюнкция. Например, в ПРЯМОЙ СИСТЕМЕ ПРОДУКЦИЙ (то есть решение строится от фактов, используя правила, и достигаются неизвестные нам цели. Здесь дизъюнкция описана к-связками (объединение связок, здесь к = 2), а конъюнкция не использует связок. Факты, например, могут быть описаны сложносочиненным предложением

$$Q(w,A) \wedge [[\sim R(v) \wedge \sim P(v)] \vee \sim S(A,v)] \quad (8)$$

Решением будет множество предложений, конъюнктивно (по умолчанию) связанных между собой) на концевых вершинах графа:

$$Q(w,A), \quad (9)$$

$$\sim R(v) \vee \sim S(A,v) , \quad (10)$$

$$\sim P(v) \vee \sim S(A,v) . \quad (11)$$

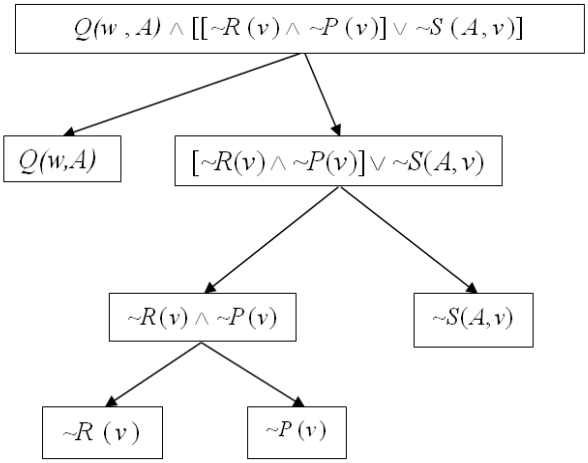


Рис. 1 – Гиперграф утверждений И/ИЛИ *(прямая продукция)*

Однако для наших целей более полезна обратная система продукций В этом случае решение строится от цели (используя правила) и согласуется с фактами, что соответствует в математике доказательству теоремы. Здесь рационально использовать гиперграфы иного вида, где конъюнкция описана k-связками, а дизъюнкция вообще не использует связок. Рассмотрим применение этого подхода на примере.

Докажем цель в следующей теореме:

Факты $R(A)$, (12)

$Q(A)$, (13)

Правила: П1: $R(y) \Rightarrow P(y)$, (14)

П2: $S(z) \Rightarrow P(B)$, (15)

Цель $P(z) \wedge Q(x)$. (16)

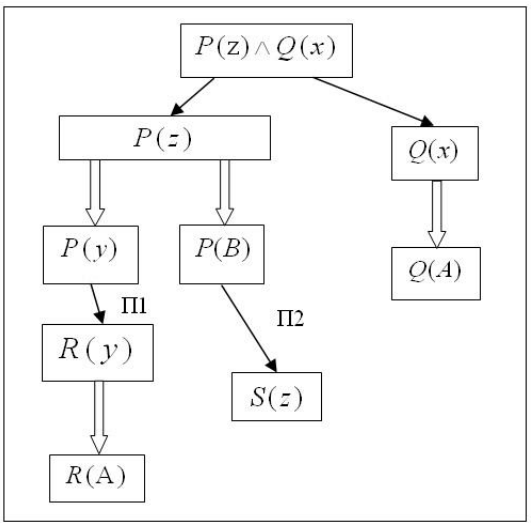


Рис. 2 – Исключение ветви графа при обратной системе продукций

По условиям теоремы две подцели $P(z)$ и $Q(x)$ должны быть непременно достигнуты. Поэтому здесь стоит к-связка (к = 2). Она связывает обе подцели в одну общую цель. В данном случае правила проще применять в виде импликации. Кроме того, правила применяются в обратном порядке, например, к литералу $P(z)$ - применяем развернутое наоборот правило $P(y) \Leftarrow R(y)$ (при условии когда $z = y$). Тогда новая подцель $R(z)$ и при $z = A$ она достигается, так как в базе данных имелся факт $R(A)$.

На правой ветви графа (Рис. 2) подцель $Q(x)$ аналогично достигается при подстановке $x = A$, так как имеется факт $Q(A)$.

Ветвь графа построенная на правиле П2 формируя новую цель $P(B)$, никак не достигается, но поскольку на общее решение это не влияет, то эта ветвь попросту отбрасывается. Решение включает левую и правую ветви графа, а среднюю ветвь, которая не согласуется – попросту удалим.

4 Переход к языку программирования «ПРОЛОГ»

Язык ПРОЛОГ – это язык «наследованный от математики» *(mathematically-derived languages).* Прологовские предложения бывают трех типов: факты, правила и вопросы.

• Факты содержат утверждения, которые являются всегда, безусловно истиной.

• Правила содержат утверждения, истинность которых зависит от некоторых условий.

• С помощью вопросов пользователь может спрашивать систему о том, какие утверждения являются истинными.

Предложения ПРОЛОГа состоят из головы и тела. Тело – это список целей, разделенных запятыми. Факты – это предложения, имеющие пустое тело. Вопросы имеют только тело. Правила имеют голову и (непустое) тело (пример, см. табл. 9).

Таблица 9 – Правила ПРОЛОГа *(пример)*

Предложения ПРОЛОГа	**Голова**	**Тело**
Факты Ф1 Ф2	 Родитель (том, боб) Родитель (боб, пат)	
Правила *пр1* *пр2*	 Предок (X, Z) :- Предок (X, Z) :-	 Родитель (X, Z). Родитель (X,Y), предок (Y, Z).
Вопрос		?- предок (том, пат).

В конце каждого факта точка. Факт - имя собственное. Переменные – имена нарицательные. Дизъюнкция – точка с запятой. Конъюнкция – запятая, имеет приоритет перед дизъюнкцией.

Сопоставление в ПРОЛОГе соответствует действию в логике, называемому унификацией. Существуют некоторые отличия от теории предикатов. Правила используют импликацию (ранее в теории предикатов от импликации уходили, здесь она остается). В ПРОЛОГе всегда применяют обратную продукцию. Рассмотрим процедуру резолюции, то есть создание нового предложения из цели G(A) и правила P(y) $\Rightarrow$ G(y). Здесь голова правила G(y), а тело правила (*условная часть*) P(y). При совпадении цели G(A) и головы правила G(y = A), обе убираются и остается только тело правила с заменой переменной (*конкретизация*). Это эквивалентно процедуре

$$G(x)\big|_{x\to y} \vee G(y) \vee P(y) \Leftrightarrow P(y). \qquad (17)$$

Кванторы представлены не явно. Комментарии: /* Это комментарий */ и % Это тоже комментарий %. Как задают вопросы? а) Простой вопрос: **? - родитель (боб, пат).** б) «Вопрос-ответ» *(это фактически доказательство теорем)*: **? - дед (альфонс, юля)**

-- --> да (18)

Программа начинает с целей и, применяя правила, подменяет текущие цели новыми, до тех пор, пока эти новые цели не окажутся простыми (*заданными*) фактами. То есть это обратная дедукция, ***позволяющая даже в условиях неуверенности в единственности решения, обеспечить выполнение именно указанной задачи.***

Процесс, в результате которого пролог-система устанавливает, удовлетворяет ли объект запросу, часто довольно сложен и включает в себя логический вывод, исследование различных вариантов и, возможно, *возвраты*. Все это делается автоматически самой ПРОЛОГ-системой и по большей части скрыто от пользователя.

Запятая между целями обозначает *конъюнкцию* целей: они *все* должны быть истинными. Однако в ПРОЛОГе возможна и *дизъюнкция* целей: должна быть истинной, *по крайней мере одна* из целей. Дизъюнкция обозначается точкой с запятой. Например: P :- Q; R. читается

так: P – истинно, если истинно Q *или* истинно R. Смысл такого предложения тот же, что и смысл следующей пары предложений**:** P :- Q и P :- R.

Порядок выполнения операций:

P :- Q, R. % P – истинно, если Q и R истинны. Из Q и R следует P %. (19)

Т.е., чтобы решить задачу P, *сначала* решите подзадачу Q, а *затем* – подзадачу R. Чтобы достичь P, *сначала* достигните Q, а *затем* R. Таким образом, различие между "декларативным" и "процедурным" прочтениями заключается в том, что последнее определяет не только логические связи между головой предложения и целями в его теле, но еще и ***порядок***, в котором эти цели обрабатываются. Всякий раз, как рекурсивный вызов процедуры вычислить приводит к неуспеху, процесс вычислений возвращается к ПРОСМОТРУ и продолжается с того предложения C, которое использовалось последним. Поскольку применение предложения C не привело к успешному завершению, пролог-система должна для продолжения вычислений попробовать альтернативное предложение.

В действительности система аннулирует результаты части вычислений, приведших к неуспеху, и осуществляет возврат в ту точку (предложение C), в которой эта неуспешная ветвь начиналась. Когда процедура осуществляет возврат в некоторую точку, все конкретизации переменных, сделанные после этой точки, аннулируются. Такой порядок обеспечивает систематическую проверку пролог-системой всех возможных альтернативных путей вычисления до тех пор, пока не будет найден путь, ведущий к успеху, или же до тех пор, пока не окажется, что все пути приводят к неуспеху.

Рассмотрим на примере, как ПРОЛОГ решает задачу, представленную в табл. 9 [20]. Система попытается достичь этой цели. Для того, чтобы это сделать, она пробует найти такое предложение в программе, из которого немедленно следует упомянутая цель. Очевидно, единственными подходящими для этого предложениями являются *пр1* и *пр2*.

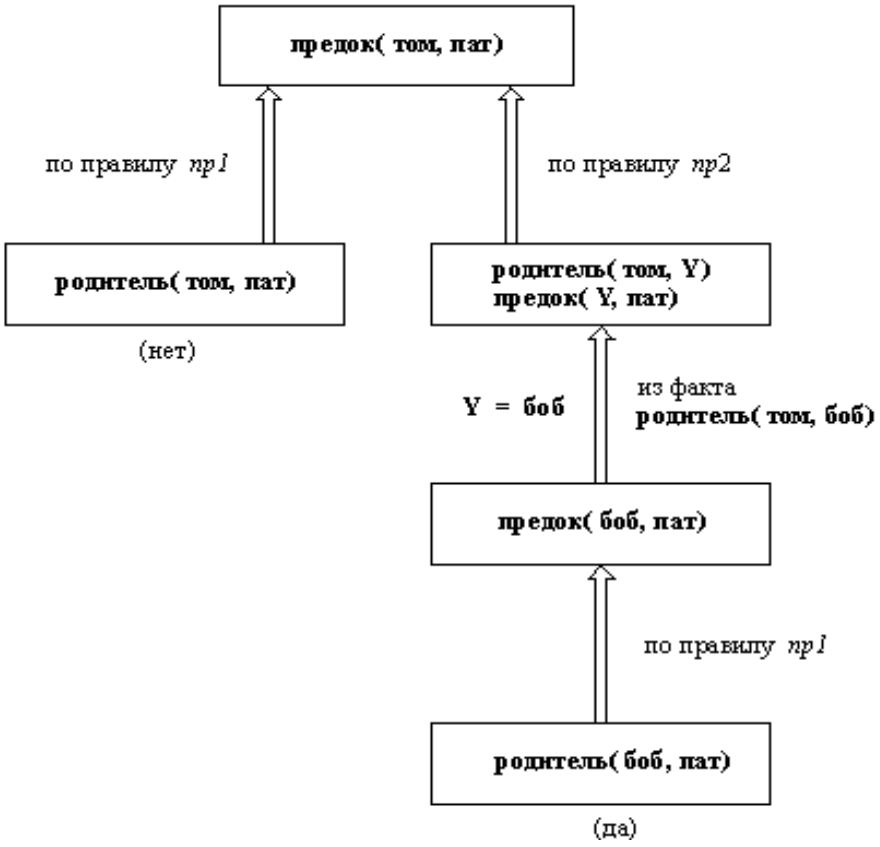


Рис. 3 – Граф задачи табл. 9 *(обратная продукция)*

Вначале система пробует предложение, стоящее в программе первым: **предок (X, Z):- родитель (X, Z)**. Поскольку цель – **предок (том, пат),** подходящие значения переменных **X = том, Z = пат**. Тогда исходная цель **предок (том, пат)** заменяется новой целью: родитель(том, пат). Однако в программе нет фактов и правила, головы которых были бы сопоставимы с целью **родитель (том, пат),** поэтому ***такая цель оказывается неуспешной***. Происходит ***возврат*** к исходной цели, чтобы попробовать второй вариант вывода цели верхнего уровня предок (том, пат).

Используем правило *пр2*. Как и раньше, переменным X и Z приписываются значения: **X = том**, **Z = пат**. В этот момент переменной Y еще не приписано никакого значения. Верхняя цель предок (том, пат) заменяется двумя целями: **родитель (том, Y), предок (Y, пат)**. Теперь перед собой *две* цели, система пытается достичь их в том порядке, каком они записаны. Достичь первой из них легко, поскольку она соответствует факту **родитель (том, боб)** из программы. Процесс установления соответствия – сопоставления (унификация) вызывает **Y = боб**. Тем самым достигается первая цель **родитель (том, боб)**, а оставшаяся превращается (из-за унификации) в **предок (боб, пат).** Для достижения этой цели вновь применяется правило *пр1*. Заметим, что это (второе) применение правила никак не связано с его первым применением. Поэтому система использует новое множество переменных правила всякий раз, как оно применяется. Чтобы указать это, мы переименуем переменные правила *пр1* для нового его применения следующим образом: **предок (X', Z') :-родитель (X', Z')**. Голова этого правила должна соответствовать нашей текущей цели **предок (боб, пат)**. Поэтому **X' = боб, Z' = пат**.

5 Заключение

Если исключить из обсуждения многочисленные победы машин в различных настольных играх, то наиболее интересным было сообщение, что экспертная система сама написала новеллу и эта новелла оказалась в числе десятка выбранных для окончательного конкурса ничего не подозревающим жюри. Правда в проекте, который длился с 2012 года, в компьютер вводились данные о сюжете, и о персонажах, а программа по специальной схеме *(разработанной командой профессора Хитоси Мацубарой из университета "Мирай" в Хакодате)* складывала уже самостоятельно все детали литературной конструкции. Хотя и здесь все как у людей. Вспомните замечательного писателя А. Дюма и в настоящее время немногих явных и множество неявных авторов многочисленных сериалов. Так что технология написания чужими руками придуманного и кратко сформулированного литературного произведения, хоть бы даже машиной, не нова.

Пока системы искусственного интеллекта способны только помочь нам решить некоторые не очень сложные задачи, где скорости реакции машин превосходят скорости реакции человек. Например, роботы в банковской системе позволяют при правильно составленной программе зарабатывать на колебаниях курсов валют и ценных бумаг. Человек вряд ли сможет с ними конкурировать, разве что в ситуациях, когда программы роботов окажутся бессильны. Некоторые виды умственной деятельности также можно переложить на плечи систем искусственного интеллекта. Но пока человеку придется самому справляться с многими задачами, ибо время роботов, способных заменить человека в многообразии жизненной деятельности еще не пришло. Внушает некоторый оптимизм только историческое объединение методов описания экспертных и нейронных систем [21] на базе нечеткой логики [22,23] и набирающий скорость прогресс в создании технических устройств элементов инфраструктуры.

В работе [24] была высказана гипотеза «необходимого развития»: любая эволюционирующая интеллектуальная система (в частности цивилизация или самообучающийся искусственный, или гибридный[1] суперинтеллект), все время усиливает свои интеллектуальные

[1] Возможность формирования планетарного интеллекта на базе глобальной сети, воздействовать на которую человечество не сможет, обсуждалось в работе [24].

ресурсы (т.е. подключаются все новые и новые блоки интеллектуальной сети) растет ее усложнение. И прежний объем обучающих задач и соответственно предложенных решений вскорости окажется недостаточным, в том смысле, что при этом возникают проблемы с формированием производных решений, растут рассогласования, которые воспринимаются как ошибки. Т.е. нарушается устойчивость картины мира. Единственным выходом здесь может быть увеличение объема известных знаний, которые обеспечивают эффективное обучение. Это увеличение может быть обеспечено наукой, расширением горизонта восприятия, доступом к новым источникам, выходом из замкнутого круга прежних понятий и представлений.

Но пока наукой в самом широком смысле вынуждены заниматься люди, пусть даже с помощью множества технических помощников. Смущает явное несоответствие скорости реакции природного нейрона и нейрона искусственной сети, то есть скорости отдельных операций вычислительных систем. Тем не менее скорость получения решения человеком частенько значительно превосходит скорость достижения результата у современных машин. Особенно в условиях одновременного решения многих задач. В чем же дело? Дело скорее всего в том, что мозг человека – это мегапроцессорная система. Одновременное подключение этого гигантского числа процессоров ускоряет получение решения в такое же, если не большее число раз. Поэтому все усилия исследователей полезно применить в области разработки мегапроцессорных систем новых поколений [24,25]. Важно также отметить, что освоение новых знаний у людей происходит порой даже непроизвольно. И когда мы хотим создать искусственную интеллектуальную систему, способную осознавать себя и понимать нас, и намерены заполнить ее базу данных и знаний, вот тут-то и возникают проблемы.

Во-первых, у человека масса знаний, которые он полагает известными (по умолчанию), машине все это надо разжевать и пояснить.

Во-вторых, заполнение базы данных машины должны делать эксперты, а их работа - высокооплачиваемая.

В-третьих, время, которое затрачивается на заполнение баз данных, а также проверку и перепроверку этого заполнения, достаточно значительное. Потому, пока не найдут эффективный машинный способ автоматического заполнения баз данных, или же не обучат эту интеллектуальную систему все это делать самостоятельно, дело быстрее не пойдет.

Самое интересное, что обнаружила цивилизация - это изучение природы и создание новых технологий, многократно усиливающих возможности людей, чтобы еще более успешно познавать мир и создавать то, что превращает жизнь в увлекательную игру. А как известно, ничего так более не привлекает людей, как игры и удовлетворение собственного любопытства, так свойственные природе человека.

Ссылки

[1] Newell A. The chess machine: an example of dealing with a complex task by adaptation / A. Newell // ACM. Proceedings of the 1955 Western joint computer conference. – 1955. – P.101 – 108.

[2] Newell A. GPS, program that simulates human thought / A. Newell, H. Simon // Defense Technical Information Center. – 1961. – Vol.4. – №10. – P. 109 – 124.

[3] Shannon C.E. A Mathematical Theory of Communication / C.E. Shannon // The Bell System Technical Journal. – 1948. – Vol. 27. – P.379 – 423; 623 – 656.

[4] Gotlob F. Izbrannye raboty / Frege Gotlob; Sost. V.V. Anashvili, A.L. Nikiforov; Per. s nem. V.V. Anashvili .– Moskva: Domintellektual, 1997. – 159 s.

[5] Turing A.M. On Computable Numbers, with an application to the Entscheidungsproblem / A.M. Turing // Proc. London Math. Soc. Ser. 2. – 1937. – Vol. 42. – P. 230-265.

[6] Newell A. Programming the Logic Theory Machine / A. Newell, F.C. Shaw // Proceedings of the Western Joint Computer Conference. – 1957. – P. 230-240.

[7] McCarthy J. Lisp 1.5 Programmer's Manual / J. McCarthy, P. Abrahams, D. Edwards et al. – MIT Press, Cambridge, Massachusetts. – 1962.

[8] Church A. Introduction to mathematical logic. Vol.1./ A. Church . – Princeton: Princeton University Press,1956. – 485 p.

[9] Robinson J. A. A Machine-Oriented Logic Based on the Resolution Principle / J. A. Robinson // Communications of the ACM. – 1965. – Vol. 12. – №1. – P. 23-41.

[10] Chang C.L. Symbolic Logic and Mechanical Theorem Proving / C.L. Chang, R.C.T. Lee. – New York: Academic Press,1973. – 331 p.

[11] Colmerauer A. Un système de communication enfrançais / Colmerauer Alain, Henry Kanoui, Robert Pasero et Philippe Roussel // Rapport préliminaire de fin de contrat IRIA, Groupe Intelligence Artificielle, Faculté des Sciences de Luminy, Université Aix-Marseille II, France, 1972.

[12] Herbrand J. Recherches sur la théorie de la demonstration / J. Herbrand // Travaux de la société des Sciences et des Lettres de Varsovie, Class III, Sciences Mathématiques et Physiques. – 1930. – Vol. 33.
[13] Pospesel H. Introdaction to Logic: Predicate Logic. Englewood Cliffs / H. Pospesel. – New Jersey: Prentice – Hall, 1976.
[14] Nil'son N. Printsipy iskusstvennogo intellekta / N. Nil'son; per. s angl. – Moskva: Radio i svyaz', 1985. – 376 s.
[15] Maslov S. An inverse method of establishing deducibility in classical predicate calculus / S. Maslov // Dokl. AN SSSR. – 1964. – Vol. 159. – P. 17–20; Proof-search strategies for methods of the resolution type // Machine Intelligence. – 1971. – N. 6. – P. 77 - 90.
[16] Van Vaalen J. An extension of unification to substitutions with an application to automatic theorem proving / J. van Vaalen // IJCAI. – 1975. – Vol.4. – P. 77–82.
[17] Sickel S. A search technique for clause interconnectivity graphs / S. Sickel // IEEE Trans. On Computers. C-25. – 1976. – № 8. – P. 823–835.
[18] Kőnig D. Theorie der endlichen und unendlichen Graphen / D.Kőnig. – Leipzig: Akademische Verlagsgesellschaf, 1936; per. s angl.: Theory of finite and infinite graphs. – Birkhäuser, 1990.
[19] Martelli A. From dynamic programming to search algorithms with functional costs / A. Martelli, U. Montanari // IJCA. – 1975. – Vol.4. – P.345-350; Optimizing decision trees through heuristically guided search // CACM. – 1978. – Vol.21. – № 12. – P. 1025–1039.
[20] Bratko I. Programmirovanie na yazyke PROLOG dlya iskusstvennogo intellekta / I. Bratko; per. s angl. – Moskva: Mir, 1990. – 560 s.
[21] Jang J.S.R. ANFIS: adaptive-network-based fuzzy inference system / J.S.R. Jang // IEEE transactions on systems, man, and cybernetics. – 1993. – Vol.23. – № 3. – P.665–685.
[22] Zadeh Lotfi A. Fuzzy sets / Lotfi A. Zadeh // Information and Control. – 1965. – Vol.8. – P. 338 – 353; Fuzzy sets and systems // System Theory; Fox J. editor. – Brooklyn, New York: Polytechnic Press, 1965. – P. 29–39.
[23] Kosko B. Fuzzy systems as universal approximation / B. Kosko // IEEE Transactions on Computers. – 1994. – Vol. 43. – № 11. – P. 1329–1333.
[24] Kuklin V. M. Vzglyad na budushchee planetarnoi tsivilizatsii / V. M. Kuklin // Universitates: Nauka i prosveshchenie. – 2003. – № 4 (16). – S. 18–22.
[25] Kuklin V. Will the artificial intelligence help us? / V. Kuklin // COMPUTER SCIENCE AND CYBERSECURITY. – 2016. – Issue 4(4). – P. 35–41 [Electronic Resource]. – Way of access: http://periodicals.karazin.ua/cscs/article/view/8266/7739.pdf.

Reviewer: Alexandr Potii, Dr. of Sciences (Engineering), Full Prof., V. N. Karazin Kharkiv National University, Kharkov, Ukraine.
E-mail: potav@ua.fm

Received: January 2017.

Author:
Vladimir Kuklin, Dr., Full Professor, head of the chair, V.N. Karazin Kharkiv National University, Kharkov, Ukraine.
E-mail: kuklinvm1@gmail.com

From mathematical logic to programming languages artificial intelligence.

Abstract. The paper considers the process of formation of the theory of expert systems on an example of formation of artificial intelligence programming language PROLOG. It showed a difficult path of awareness of artificial intelligence and the motives that led to the emergence of expert systems that are based on mathematical logic. We discuss the basic ideas and procedures that led to the construction of the first department of mathematical logic - predicates theory, representation of procedure on hypergraphs, and then to create a Prolog language. Progress has been made in the development of intelligent systems and the problems faced by researchers are discussed.

Keywords: Theory predicates, hypergraphs AND/OR, Prolog programming.

Рецензент: Олександр Потій, д.т.н., проф., Харківський національний університет імені В.Н. Каразіна, м. Харків, Україна.
E-mail: potav@ua.fm

Надійшло: Січень 2017.

Автор:
Володимир Куклін, д.ф.-м.н., проф., завідувач кафедри, Харківський національний університет імені В.Н. Каразіна, Харків, Україна. E-mail: kuklinvm1@gmail.com

Від математичної логіки до мов програмування штучного інтелекту.

Анотація. Розглянуто процес становлення теорії експертних систем на прикладі формування мови програмування штучного інтелекту ПРОЛОГ. Показаний складний шлях усвідомлення проблем штучного інтелекту і мотиви, які привели до появи експертних систем, побудованих на основі математичної логіки. Обговорюються основні ідеї і процедури, які привели до побудови спочатку відділу математичної логіки - теорії предикатів, поданням процедур цієї теорії на гіперграфах і потім до створення мови ПРОЛОГ. Відзначається прогрес у розвитку інтелектуальних систем та проблеми, що стоять перед дослідниками.

Ключові слова: теорія предикатів, гіперграфи І/АБО, мова програмування ПРОЛОГ.