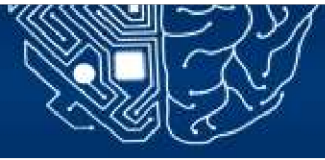
СПбГЭТУ «ЛЭТИ»

ПЕРВЫЙ ЭЛЕКТРОТЕХНИЧЕСКИЙ

Логические модели и логическое программирование

Конспект лекции

СПбГЭТУ «ЛЭТИ», 2022 г.



1 ВВЕДЕНИЕ

Одним из способов представления знаний в информационных системах является язык математической логики, позволяющий формально описывать понятия предметной области и связи между ними.

В отличие от естественного языка, который очень сложен, язык логики предикатов использует только такие конструкции естественного языка, которые легко формализуются. Логика предикатов – это языковая система, которая оперирует с предложениями на естественном языке в пределах синтаксических правил этого языка. Язык логики предикатов использует слова, которые описывают:

– понятия и объекты изучаемой предметной области;

– свойства этих объектов и понятий, а также их поведение и отношения между ними.

В терминах логики предикатов первый тип слов называется термами, а второй тип слов – предикатами. Термы представляют собой средства для обозначения интересующих нас индивидуумов, а предикаты выражают отношения между различными индивидуумами, которые обозначаются с помощью термов.

Логическая модель представляет собой множество предложений, выражающих различные логические свойства именованных отношений.

Логическое программирование – это подход, при котором пользователь описывает предметную область совокупностью предложений в виде логических формул, а кибернетическое устройство, манипулируя этими предложениями, строит необходимый для решения задач вывод.

1.1 Простейшие конструкции языка предикатов

Одной из простейших конструкций логики предикатов является терм. Терм - это знак (символ) или комбинация знаков (символов), являющаяся наименьшим значимым элементом языка. К термам относятся константы, переменные и функции. Константы обозначают конкретные объекты реального мира. Переменные используются для обозначения некоторого из возможных объектов реального мира или их совокупности. Обычно их обозначения начинаются с заглавной буквы.

Структуры – представляют собой последовательность разделенных запятыми констант или переменных, которые заключены в круглые скобки и следуют за функциональным символом (функтором). Функторы обозначают операторы, которые после воздействия на объект возвращают некоторое значение. Пример записи структур: сумма (1,2); +(1,2); удвоить (X), а примерами работы с ними и их унификации могут быть:

?- +(2, 3) = 2 + 3. true.

?- +(2, *(5,6)) = +(X, *(5, Y)). X = 2, Y = 6.

?- Result is +(2, *(5,6)). Result = 32.

Предикат – это логическая функция, которая выражает отношение между своими аргументами и принимает значение «истина», если это отношение имеется, или «ложь», если оно отсутствует. Заключенная в скобки последовательность из n термов, перед которой стоит предикатный символ, называется n-местным (или n-арным) предикатом, который принимает значения «истина» или «ложь» в соответствии со значением термов, являющимися его аргументами.

Примеры записи предикатов:

является (ласточка, птица).

отец (X, ”Петр”).

Такого типа предикаты получили название атомарных предикатов и соответствуют наиболее простым предложениям разговорного языка – нераспространенным предложениям. В обычном человеческом языке из нераспространенных предложений с помощью соединительных союзов, местоимений, и других частей речи строят более сложные конструкции – сложные предложения.

1.2 Предикатные формулы

В логике предикатов сложным предложениям естественного языка соответствуют предикатные формулы. Предикатные формулы образуются из атомарных предикатов с использованием логических связок. Эти связки еще называют пропозициональными и выделяют пять типов таких связок, которые приведены в таблице 1.

Таблица 1. Обозначение и назначение логических связок

Обозначение	¬	*, или ^*	*; или ∨*	->	*<-> или ~*
Читается, как …	НЕ	И	ИЛИ	Если …, То , Из … следует	Тогда и только …
Операция	отрицание	конъюнкция	дизъюнкция	импликация	эквиваленция

В логических выражениях порядок выполнения логических операций задается круглыми скобками. При их отсутствии логические связки имеют следующий приоритет использования: сначала выполняется отрицание ("не"), затем конъюнкция ("и"), после конъюнкции — дизъюнкция ("или") и в последнюю очередь — импликация.

В логике высказываний показано, что операцию импликации можно выразить через дизъюнкцию и отрицание: A -> B = ¬A ∨ B, а операцию эквивалентности можно представить через операции отрицания, дизъюнкции и конъюнкции: A <-> B = ¬A ∨ B) ^ (¬B ∨ A).

Таким образом, операций отрицания, дизъюнкции и конъюнкции вполне достаточно, чтобы описывать и обрабатывать любые логические высказывания. Что касается логического программирования, то в нем наиболее часто используют связки «И», «НЕ» «ЕСЛИ». Тогда предикатная формула, соответствующая сложному предложению, может иметь вид, например,

является(ласточка, птица) <- имеет (ласточка, крылья),

владеет (ласточка, гнездо).

где является/2, имеет/2, владеет/2 – это атомарные предикаты, а «,» и «<-» представляют собой логические связки.

Однако приведенная конструкция предикатной формулы позволяет делать утверждение не только о конкретном индивидууме, которым является ласточка, но и обо всех индивидуумах из класса птиц, используя для этого вместо констант переменные:

является(X, птица) <- имеет(X,крылья), владеет(X, гнездо)

Используя переменные вместо конкретных имен, мы приходим к более общим понятиям кортежа длины n, предиката и логической формулы. Однако предикат, который содержит переменные, например,

имеет (X, крылья)

не может быть однозначно оценен, так как невозможно определить ложь он или истина, Его значение определяется после подстановки в переменную некоторой константы.

Однако, иногда можно определить значения предиката, не делая подстановок, используя кванторы общности ($\forall$) и существования ($\exists$), которые обозначают «для всех» и «существует, по крайней мере, одно».

Тогда приведенная выше логическая формула будет записана в виде:

($\forall$ X) [является (X, птица) <- имеет (X, крылья), владеет (X, гнездо)]

и соответствуют предложению, которое может читаться как: «любой X является птицей, если этот X имеет крылья и владеет гнездом».

Кванторное логическое высказывание с квантором всеобщности вида $\forall$ xA(x) – истинно только тогда, когда для каждого объекта x из заданной совокупности высказывание A(x) истинно. Логическое высказывание с квантором существования $\exists$ xA(x) – истинно только тогда, когда в заданной совокупности существует объект x, такой, что высказывание A(x) истинно.

Кванторы $\forall$ и $\exists$ могут использоваться и для любого числа переменных, а при их комбинации очень важен порядок их использования. Рассмотрим их различное использование, и соответствующее семантическое толкование высказываний на примере использования двуместного предиката любит/2, который описывает отношение «X любит Y»:

$\forall$ X $\forall$ Y любит (X, Y) – все любят всех;

$\exists$ X $\forall$ Y любит (X, Y) – существует такой X, который любит всех;

$\forall$ X $\exists$ Y любит (X, Y) – любой X любит хотя бы одного Y;

$\exists$ X $\exists$ Y любит (X, Y) – существует такой X, который кого-либо любит;

$\exists$ Y $\exists$ X любит (X, Y) – существует такой Y, которого кто-то любит;

$\forall$ Y $\exists$ X любит(X, Y) – каждого Y любит хотя бы один X.

1.3 Определение правильно построенной формулы

Комбинируя логические связки и кванторы можно рекурсивно определить составную формулу логики предикатов, называемую правильно построенной формулой (далее просто ППФ или логическая формула). Если говорить применительно к естественному языку, то ППФ описывает обычное предложение общего вида:

1. Термом является либо константа, либо переменная, либо кортеж из n термов, перед которым стоит функтор.

2. Предикат – кортеж из n термов, перед которым стоит предикатный символ.

3. Атомарный предикат является логической формулой.

4. Если F и G - логические формулы, то (F), (F,G), (F$\vee$G), ($\neg$F), (F$\rightarrow$G), (F$\leftrightarrow$G) – также являются логическими формулами.

5. Если F(X) - логическая формула, то выражения $\forall$ X F(X) и $\exists$ X F(X) является логическими формулами.

6. Все результаты, получаемые повторением конечного числа раз пунктов с 1 по 6, являются правильными логическими формулами.

Множество предложений, которые построены согласно этим правилам, образуют язык логики предикатов первого порядка, который, являясь формализованным аналогом обычной логики, дает возможность строго рассуждать об истинности и ложности утверждений и об их взаимосвязи. В частности, о логическом следовании одного утверждения из другого, или, например, об их эквивалентности.

Рассмотрим классический пример формализации утверждений естественного языка в логике первого порядка. Возьмем рассуждение: "Каждый человек смертен. Сократ – человек. Следовательно, Сократ смертен". Обозначим:

"x есть человек" – через ЧЕЛОВЕК(x),

"x смертен" – через СМЕРТЕН(x).

Тогда утверждение "каждый человек смертен" может быть представлено формулой:

($\forall x$)(ЧЕЛОВЕК(x) $\rightarrow$СМЕРТЕН(x)).

А утверждение "Сократ – человек" – формулой:

ЧЕЛОВЕК(Сократ),

и "Сократ смертен" – формулой:

СМЕРТЕН(Сократ).

Утверждение в целом теперь может быть записано формулой:

$(\forall x)$(ЧЕЛОВЕК(x) $\rightarrow$СМЕРТЕН(x)) & ЧЕЛОВЕК(Сократ) $\rightarrow$СМЕРТЕН(Сократ).

2 ЛОГИЧЕСКИЙ ВЫВОД

Логический вывод – это процесс получения из множества правильно построенных логических формул {Si} некоторой новой ППФ (s) путем применения одного или нескольких правил вывода.

Одним из таких правил является правило modus ponens, которое используется в исчислении высказываний. Его еще называют правилом отделения или гипотетическим силлогизмом. Простой силлоги зм — это рассуждение мысли, которое состоит из трёх простых атрибутивных высказы-

ваний: двух посылок и одного заключения. Примером силлогизма может быть только что рассмотренный пример:

Всякий человек смертен (большая посылка)

Сократ — человек (меньшая посылка)

Сократ смертен (заключение)

Правило вывода modus ponens позволяет от утверждения условного высказывания A$\rightarrow$B и утверждения его основания A (антецедента) перейти к утверждению следствия B (консеквенту). Другими словами, если A и A$\rightarrow$B — выводимые формулы, то B также выводима. Формат записи этого правила вывода будет иметь вид:

A, A $\rightarrow$ B B

Следует отметить, что рассмотренное правило является частным случаем правила резолюций, которое в свою очередь относится к методу доказательства теорем через поиск противоречий. Правило резолюций, последовательно применяемое для списка резольвент, позволяет ответить на вопрос, существует ли в исходном множестве логических выражений противоречие.

2.1 Правило резолюции для простых предложений

Наиболее простой метод логического вывода использует только одно правило вывода, называемое резолюцией, которое применяют к логическим формулам вида:

факт: A

отрицание: ¬(A1, ... , An)

импликация: A ← B1, ... , Bm ,

где Ai (i = 1, n) и Bj (j = 1, m) - произвольные предикаты.

Рассмотрим наиболее простую из форм резолюции для случая всего лишь двух правильно построенных формул S = { S1 , S2 } вида:

S1 (отрицание): ¬ A

S2 (импликация): A ← B ,

в которых предикат A из S1 совпадает с предикатом A левой части S2. В результате одного шага вывода из S1 и S2 будет получена новая ППФ вида:

S (резольвента): ¬B

На этом шаге вывода правильно построенные формулы S1 и S2 называются родительскими предложениями, а S – резольвентой, которая получается в результате применения резолюции к S1 и S2.

Резолюция в этом простейшем случае соответствует правилу вывода modus tollens, которое записывается в виде:

¬A, A ← B

_____

(*) ¬B

и соответствует следующему умозаключению:

Допуская, что: не A

и A если B

_____

Выводим: не B

В еще более простом случае, когда логическая формула S1 представляет собой отрицание, а логическая формула S2 – факт:

S1 (отрицание): ¬A

S2 (факт): A

применение правила резолюции даст резольвенту в виде пустого отрицания

S : □ ,

которое означает противоречие. Этот шаг вывода может быть записан в следующем виде:

¬A, A

_____

□

и соответствует следующему умозаключению:

Допуская, что: не А

и А

_____

Выводим противоречие.

2.2 Правило резолюции для сложных предложений

Реальные логические модели содержат значительно более сложные предложения. Так отрицание могут содержать несколько предикатов, так же как и правые части импликаций. Поэтому более общим является случай, когда родительские предложения имеют вид:

S1: ¬(A1,... ,Ak, ...,An)

S2: Ak ← (B1, ...,Bm) (где 1<k<n)

Здесь некоторый предикат Ak из отрицания S1 совпадает с предикатом левой части S2. В этом случае один шаг вывода заменяет предикат Ak в логической формуле S1 на правую часть логической формулы S2 и в качестве резольвенты получают отрицание вида:

S: ¬(A1, ..., Ak-1, B1,..., Bm, Ak+1, ..., An)

Данное правило вывода для сложных предложений проиллюстрируем содержательным примером. Например,

допуская, что Не (темно и зима и холодно)

и что Зима если Январь

_____

выводим, что НЕ (темно и январь и холодно)

Рассмотрим случай, когда существует две логические формулы, в которых S1 имеет тот же вид, что и ранее, а логическая формула S2 представляет собой факт:

S1: ¬ (A1,... ,Ak, ...,An)

S2: Ak

Причем факт Ak аналогичен одному из предикатов, входящих в S1. Тогда один шаг вывода заключается в том, что он только вычеркивает предикат Ak из S1 и получает резольвенту вида

S: ¬(A1, ..., Ak-1, Ak+1, ..., An)

и данный шаг вывода можно иллюстрировать следующим примером

допуская, что Не (темно и зима и холодно)

и что Зима

выводим, что НЕ (темно и холодно)

2.3 Простая резолюция сверху вниз

Рассмотренные выше правила применяются на каждом шаге вывода только к двум родительским предложениям. Вместе с тем описание любой области знания содержит множество ППФ. Рассмотрим процесс логического вывода для примера, когда знания выражаются двумя предложениями.

S2: получает(студент, стипендию) ←
сдает(успешно, сессию, студент)

S3: сдает(успешно, сессию, студент)

Задача, которую надо решить на этом множестве ППФ состоит в том, чтобы ответить на вопрос: ”Получает ли студент стипендию?”. Когда будет использоваться обычная система логического вывода, то такой вопрос представляется запросом в виде отрицания:

S1: ¬ получает(студент, стипендию)

и задача системы состоит в том, что она должна отвергнуть это отрицание при помощи других предложений, демонстрируя, что данное допущение ведет к противоречию. Этот подход часто применяется в математике и называется доказательством от противного.

Теперь представим себе, что исходная логическая модель, составленная из трех предложений S1, S2 ,S3 поступает на вход системы логического вы-вода компьютерной системы.

Шаг 1. Система на первом шаге применит правило вывода (*) к родительским предложениям S1 и S2 и получит резольвенту:

s: ¬ сдает(успешно, сессию, студент)

Шаг 2. Используя правило (**) к ППФ s и S3 система выводит противоречие, а именно s':

Таким образом, для доказательства противоречивости S1, S2 и S3 оказалось достаточно двух шагов вывода. Если считать, что S2 и S3 не противоречат друг другу, то они совместно противоречат S1 или другими словами подтверждают предложение: получает (студент, стипендию). И в этом случае ответом на исходную задачу является «Истина (true/ДА)».

Логический вывод, который порождает последовательность отрицаний, такую как s1 , s , s' в данном примере, называется резолюцией сверху вниз.

2.4 Общая резолюция сверху вниз

Однако в большинстве случаев структура предложений имеет более сложный вид. В частности, предикаты и логические формулы в качестве термов могут содержать не только константы, но и переменные, и функции. В этих условиях несколько модифицируется и сама процедура логического вывода. Чтобы получить самые общие представления об общей резолюции сверху вниз рассмотрим для примера два родительских предложения вида:

S1: ¬ получает(студент, У)

S2: получает(X,стипендию) ← сдает(успешно, Z, X)

К ним непосредственно уже нельзя применить правило резолюции, так как они не содержат одинаковых предикатов в левой части импликации и в отрицании. Данные предложения содержат три переменных X, Y, Z, которые

неявно универсально квантифицированны. Рассмотрим первое предложение S1, которое утверждает, что:

для всех Y (студент не получает Y)

Причем выражение «для всех» понимается как «для всех индивидуумов, из какой либо области, выбранной для интерпретации предложений». При интерпретации S1 и S2, по крайней мере, один индивидуум Y будет связан с именем «стипендия» и поэтому непосредственным следствием S1 является более конкретное предложение:

S11: ¬ получает(студент, стипендию)

Аналогично рассматривается S2 на области интерпретации S1 и S2 и, выбирая для X, индивидуум с именем «студент», получаем более конкретное предложение:

S21: получает(студент, стипендию) ←

сдает(успешно, Z, студент)

Теперь имеем два предложения S11 и S21, которые удовлетворяют условию применимости правила резолюции. Это условие предполагает наличие одинаковых предикатов в левой части импликации и в отрицании. Поэтому резольвентой логических формул S11 и S21, после применения к ним правила (*) будет:

s: ¬ сдает(успешно, Z, студент)

Предикат получает (студент, стипендию) называется общим примером родительских предикатов

получает(X, стипендию)

получает(студент, У)

и получен с помощью унификатора вида Θ = {X:= студент, У:= стипендия}.

2.5 Унификаторы и примеры унификации

Унификатором называется множество присваиваний вида

Θ = {X1:= t1, ..., Xn:= tn}

где Xi - переменная, а ti – терм, применение которых к двум выражениям дает одинаково общие примеры.

На практике унификаторы определяют, сравнивая по очереди соответствующие аргументы предикатов и выписывая те присваивания

термов переменным, которые сделали бы эти аргументы одинаковыми. Для пояснения этого процесса рассмотрим ряд примеров унификации, которые приведены в таблице 2.

Таблица 2. Примеры унификации

Родительские предложения	Унификатор	Общий пример
p(5), p(5)	Θ- пустое множество (не заменяется ни одна переменная)	p(5)
p (x), p(5)	Θ= {x:=5}	p(x) Θ = p(5) Θ = p(5)
p(x), p(y)	Θ = {x:=y}	p(y)
p(x, y), p(5, x)	Θ = {x:=5, y=x} = = {x:=5, y:=5}	p(x,y) Θ = p(5,x) Θ = p(5,5)
¬ p(5, x) p(x, y) ←q(x)	Θ = {x:=5, y:=5}	p(5,5) резольвента: s: ¬q(x) Θ = ¬ q(5)

2.6 Решение задач и извлечение ответа

Решение задачи с использованием логического программирования разбивается на 3 этапа:

На первом этапе необходимо сформулировать наши знания и допущения о предметной области в виде множества ППФ.

На втором этапе нужно выразить конкретную задачу, поставленную на предметной области, как запрос об одном или нескольких отношениях. Обычно запрос ставится как исходное отрицание.

Составлением допущений и исходного запроса завершается работа программиста, цель которой – сформулировать задачу.

Третий шаг выполняется компьютером, который пытается решить задачу, строя доказательство от противного. Он строит вывод сверху вниз, начиная с исходного отрицания, и порождает последовательность отрицаний D1, D2, ... , Dn. Если может быть построен вывод, который заканчивается отрицанием, то есть Dn = □ (противоречие), то этот вывод, называемый успешным и сразу дает решение поставленной задачи.

В качестве примера рассмотрим последовательность выполнения всех этих этапов на примере вычислительной задачи нахождения значения факториала некоторого числа.

На первом этапе необходимо в виде ППФ сформулировать наши знания о вычислении факториала, которые можно представить двумя предложениями:

S1 : факториал(0,1)

S2 : факториал(X,F) ← X>0, N=X-1,факториал(N,Y), F=Y*X

На втором этапе конкретную задачу, например, вычисление значение факториала числа 3 (3!=1*2*3), формулируем в виде запроса как исходного отрицания:

D1: ¬ факториал(3, Z)

На третьем этапе система логического вывода выполнит приведенную в таблице 3 последовательность действий, применяя на каждом из шагов соответствующие правила резолюции.

Таблица 3. Вывод на основе резолюции

Шаг	Родительские предложения	Унификатор	Отрицание (резольвента)
1	D1, S2	Θ = {X:=3, F:=Z, Y:=F/X}= = {N:=2 , Y:= Z/3}	D2: ¬ факториал(2, Z/3)
2	D2, S2	Θ = {X:=2, F:=Z/3, Y:=F/X}= = {N:=1 , Y:= Z/6}	D3: ¬ факториал(1, Z/6)
3	D3, S2	Θ = {X:=1, F:=Z/6, Y:=F/X}= = {N:=0 , Y:= Z/6}	D4: ¬ факториал(0, Z/6)
4	D4, S1	Θ = {Z/6:=1} = {Z:=6}	□ (противоречие)

Полученное на шаге 4 противоречие подтверждает отрицание D_1, а стало быть, вывод является успешным и дает решение: факториал(3,z), с унификатором Θ = {Z:=6}. Ответ, который выдаст система логического вывода, будет: Z:=6.