1 Теоретическая часть

1.1 Общее

Конъюнкция, дизъюнкция, отрицание – базовые функции матлогики. Предикат - логическая функция. Базис пролога – матлогика.

В прологе используется символьная обработка. Декларативная методология. Мы описываем систему знаний из предметной области. Потом задаем вопрос, но хотим получить не только да/нет, но и как (как побочный эффект)? Не запрещено использовать символы.

Запросы могут быть конъ или дизъ, но нам будут запрещать их использовать

что такое декларативно? (интернет - Декларативное программирование — это парадигма, при которой описывается желаемый результат, без составления детального алгоритма его получения.)

1.2 Терм

Терм - основной элемент языка Prolog. Терм – это:

1. константа – ква(используется для обозначения объекта/процесса предметной области или для обозначения конкретного отношения):
   - число (целое, вещественное),
   - символьный атом – комбинация символов латинского алфавита, цифр и '_' (символа подчеркивания), начинающаяся со строчной буквы,
   - строка – последовательность символов, заключенных в кавычки;
2. переменная:
   - именованная – комбинация символов латинского алфавита, цифр и '_', начинающаяся с прописной буквы или символа подчеркивания, может связываться с различными объектами (конкретизироваться),
   - анонимная - обозначается символом '_', не может быть связана со значением;
3. составной терм (составное тело) – средство фиксации информации о том, что между объектами существует определенная связь. Синтаксически представляется так: f(t1, t2, ..., tm), где f - главный (тк внутри мб другие)

функтор – символьная константа, обозначающая имя отношения между объектами; t1, t2, ..., tm – термы (в том числе и составные), являющиеся аргументами (арность – число аргументов).

student(ivanov, mgtu) - константы student(X, mgtu) – группа студентов из мгту. В момент фиксации система не знает, что такое X student(ivanov, mgtu) и student(ivanov) – для системы разные запросы.

Первые аргуметы считаются как объекты одной природы, вторые - другой. Только мы определям смысл.

1.3 Переменные

Зачем нужны переменные – для повышения уровня абстракции. Чем больше переменных, тем выше уровень абстракции.

Именованная переменная представляет собой комбинацию символов латинского алфавита, цифр и '_', начинающуюся с прописной буквы или символа подчеркивания. В процессе выполнения программы именованные переменные могут связываться с различными объектами – конкретизироваться. Именованная переменная является уникальной в рамках предложения. В разных предложениях может использоваться одно имя переменной для обозначения разных объектов.

Анонимная переменная обозначается символом '_'. Она не может быть связана со значением. Любая анонимная переменная уникальна.

(Уникальность: именованная переменная уникальна в рамках одного предложения. Анонимная переменная уникальна всегда.)

(Именованные) Переменные предназначены для передачи информации «во времени (за конечное число шагов получить результат) и пространстве (передача значений через параметры)». Для этого в какой-то момент переменная дб конкретизирована каким-то значением. Но это может быть ошибочным, есть механизм отказа (отката), реконкретизировать переменную. В момент фиксации система не знает, какой объект представляет переменная. Во многих языках последовательность действий при работе с переменными такая: задать значение переменной, а затем работать с самой переменной. В прологе же особый способ работы с переменными. Методом проб и ошибок. Цель – подтвердить истинность вопроса с помощью бз.

Установление значения длля переменной не связано с понятием типа (по указателям).

Анонимные переменные система не конкретизирует значениями.

Именованная переменная входит в факты и правила с квантором всеобщности, а в вопрос с квантором существования.

1.4 Что собой представляет программа на языке пролог?

Программа на Prolog представляет собой базу знаний и вопрос.

- **База знаний** состоит из предложений – фактов и правил, – используя которые программа выдает ответ на вопрос. Каждое предложение должно заканчиваться точкой.
  - **Правило** имеет вид: A :- B1, ... , Bn, где A – заголовок правила (составной терм, который содержит знание); B1, ... , Bn - тело правила (составные термы, которые содержат условия истинности этого знания), символ " : —" – это специальный символ-разделитель. Заголовоок - фиксацияя знания о том, что между аргументами мб истинная связь
  - **Факт** – это частный случай правила – предложение, в котором отсутствует тело (то есть тело пустое).
- **Вопрос** – ??это частный случай правила – предложение, которое состоит только из тела. Используется, чтобы определить, выполняется ли некоторое отношение между описанными в программе объектами. Система рассматривает вопрос как цель, к которой (к истинности которой) надо стремиться. Ответ на вопрос может оказаться логически положительным или отрицательным, в зависимости от того, может ли быть достигнута соответствующая цель.

Вопросы и правила (факты) без переменных – основные (предназначены для описания отношений, формирования базы знаний), с переменными – неосновные (для поиска ответа в базе знаний)

В базе знаний нет порядка.

В момент фиксации знания (если еще оно с условием, переменными) – условная истина.

Алгоритм унификации - единственный алгоритм доказательства. Многократно запускается (в какой момент?)

1.5 Структура программы на Prolog

Программа на Prolog состоит из следующих разделов, каждый из которых начинается со своего заголовка.

- директивы компилятора — зарезервированные символьные константы,

- CONSTANTS — раздел описания констант,
- DOMAINS — раздел описания доменов,
- DATABASE — раздел описания предикатов внутренней базы данных,
- PREDICATES — раздел описания предикатов,
- CLAUSES — раздел описания предложений базы знаний,
- GOAL — раздел описания внутренней цели (вопроса).

В программе не обязательно должны быть все разделы.

1.6 Как реализуется программа на Prolog? Как формируются результаты работы программы?

Ответ: Программа на Prolog представляет собой базу знаний и вопрос. База знаний состоит из предложений – фактов и правил, которые задают истинные знания. Ответ на вопрос может оказаться логически положительным или отрицательным, в зависимости от того, может ли быть достигнута соответствующая цель.

Вопрос рассматривается системой как цель: найти возможность, исходя из базы знаний, ответить «Да» на поставленный вопрос . Вариантов ответить «Да» на может быть несколько. При поиске ответа рассматриваются альтернативные варианты и находятся все возможные решения (методом проб и ошибок) - множества значений переменных, при которых на поставленный вопрос можно ответить - «Да».

Для выполнения логического вывода используется механизм унификации, встроенный в систему.

Лекция 2

Алгоритм унификации - единственный алгоритм доказательства. Многократно запускается (в какой момент?) <3

процедурные и декларативные особенности пролога

Чтобы ответить на вопрос, надо подобрать знание, сравивая вопрос с знаниями. И то, и то – составной терм. Сравнивает по 2 состсавных терма по формальному признаку. Порядок формально установлен сверху вниз.

Если есть переменная и в вопросе, и в формулировке знания (а если еще и на одной позиции)

1.7 Что такое предикат в матлогике (математике)?

Предикат в математической логике – это (логическая) функция со множеством значений 0, 1 (истина/ложь), определенная на некотором множестве параметров. Предикат называю n-арным, если он определен на n-ой декартовой степени множества M. Таким образом, каждый набор параметров характеризуется либо как «истинный», либо как «ложный».

1.8 Что описывает предикат в Prolog?

Процедура – совокупность правил (описывающих определенное отношение), заголовки которых имеют одинаковые главнаые функторы, одинаковое число аргументов, обозначающих объекты одной природы (не про память).

Это одно знание, которое мб зафиксировано через несколько. Структура знания описывается в разделе предикатов.

??Предикат – отношение, определяемое процедурой. Таким образом, предикат в Prolog описывает отношение между аргументами процедуры.

подстановки и примеры терма

Дан неосновной термы: A(X1, ..., Xn), Xi – переменные.

Чтобы подбирать значения переменных нужно построить подстановку. Принято обозначать Θ

Подстановкой называется множество пар вида xi=ti, где xi – переменная, ti – терм, не содержащий переменных (ti – значение для переменной xi).

Применение подстановки заключается в замене каждого вхождения переменной x_i на соответствующий терм.

Пусть $\Theta : X_1 = t_1, X_2 = t_2, \ldots, X_n = t_n$ – подстановка. Тогда результат применения подстановки к терму обозначается $A\Theta$.

Терм B называется примером терма A, если существует такая подстановка Θ, что $B = A\Theta$.

Терм C называется общим примером термов A и B, если существуют такие подстановки Θ_1 и Θ_2, что $= A\Theta_1$ и $= B\Theta_2$.

A = plus(1, 2, z) plus(X, Y, 3)

В процессе выполнения программы система, используя встроенный алгоритм унификации, пытается обосновать возможность истинности вопроса, строя подстановки и примеры термов (вопроса и формулировки знания), используя базу знаний. Построение и подстановки производится путём конкретизации переменных. Сами термы хранятся в стеке.

Простейшие правила логического вывода

Правила вывода – утверждения о взаимосвязи между допущениями, которые с позиции исчисления предикатов верны всегда. Возможны 4 варианта:

1. факты основные (квантор существования), вопрос основной – правило: совпадение
2. факты основные (квантор существования), вопрос неосновной – правило: обобщение факта
3. факты неосновные (квантор всеобщности), вопрос основной – правило: конкретизации факта
4. факты неосновные (квантор всеобщности), вопрос неосновной – система должна посстроить пример терма-вопроса и терма-знания (подобрать соответствующие подстановки). Общий пример строится в 2 шага – сначала конкретизация правила, а потом правило обобщения.

Унификация терма

Подбор знания. Назначение – если сработал, значит для доказательства истинности ответа на вопрос можно использовать знание.

По императивному принципу

Унификация – операция, которая позволяет формализовать процесс логического вывода. С практической точки зрения - это основной вычислительный шаг, с помощью которого происходят:

- двунаправленная передача параметров процедурам (знание в несколько предложений),
- неразрушающее присваивание (конкретизация),
- проверка условий (доказательство).

В процессе работы система выполняет большое число унификаций. Попытка "увидеть одинаковость"– сопоставимость двух термов, может завершаться успехом или тупиковой ситуацией (неудачей). В последнем случае включается механизм отката к предыдущему шагу.

Унифицировать (понять, что это знание подходит для доказательства этого вопроса) два терма. Два терма про одно и то же (T1=T2: = – принудительный (явный) запуск унификации)

Два терма унификируются по следующим правилам:

1. Если T1 и T2 – константы: только если они совпадают
2. Если T1 – неконкретизированная переменнаяя, а T2 – константа или составной терм, не содержащий в качестве аргумента T1: унификация успешна, а T1 конкретизируется значением T2
3. Если T1 и T2 – неконкретизированные (не имеющие значения) переменные: унификация всегда успешна, причем T1 и T2 становятсяя сцеплёнными двумя именами (указателями) одного и того же объекта.
   Если одна из переменных или оин из тером конкретизивуется значением, то второй моментально тоже конкретизируется им же
4. Если T1 и T2 – составные термы (например, вопрос и заголовок): успешно унифицируются если
   (a) у T1 и T2 одинаковые главные функторы

(b) T1 и T2 имеют равные арности

(c) успешно унифицируется каждая пара их соответствующих компонент

Алгоритм унификации

рабочее поле, стек (для хранения равенств, возможность равенства которых проверяется), результирующая ячейка (динамическая облсьб памяи, в которой накаплливается подставновка)

T1=T2

Начало

- Занеси T1=T2
- Неудача=0
- Пока стек не пус
  - Считать из стека в рабочее поле S=T
  - обработать
    - а)если S и T – несовпадающие консаны, то неудача
    - б) совпадающие константы – на конец цикла
    - в) S – переменная, а T – терм, содержащий S, то неудача=1 и выход из цикла
    - г) S – переменная, а T – терм, не содержащий S, (для S это возм знач), то отыскать в стеке и результ ячейке все вхождения S и заменить их на T. Добавить в результирующую ячейку S=T, следующий шаг цикла.
    - д) S и t составные термы, им разные функторы или разную арность, то неудача=1 и выхоод из циклы
    - е) составные термы с одинаковыми функторами и арностями, занести в стек равенства S1=T1, ... Sk=Tks
  - очистить рабочее поле
  - конец цикла
- если неудача=1, то унификация невозможна, иначе – успешна, а РЯ содержит подстановку – эта подстановка называется наиболее общий унификатор.

Терм S называется более общим, чем T, если T является примером S, а S не является примером T.

Терм S называется наиболее общим примером T1 и T2, если S – такой их общий пример, который являяется более общим по отношению к любому другому их примеру.

Унификатором двух термов называется подстановка, которая будучи применена к 2 термам даст одинаковый результат.

Наиболее обзим унификатором двух ттермов называется унификаттор, соответсв наиболее обзему примеру двух термов.

Если 2 терма унифицируемы, то сущ единственный (с ттчностью до переименования переменных) наиболее общзий унификатор, а значчит наиболее общий пример.

Алгоритм унификации и строит наиболее общий пример

....

1.9 Общая схема согласования целевых утверждений

Рещение задачи с пом лог прог начинается с задания вопроса G и завершается получением одного из двух результатов:

- Успешного согласования вопроса и программы, а в качестве побочного эффекта будет получена подстановка, которая содержит значения переменных, при которых вопрос является примером программы (или несколько)
- Неудача

Вычисления с пом конеч лог прог представляет собой пошаговое преобразование исходношо вопроса.

На каждом шаге имеется некоторая конъюнкция целей, выводимость которой следует доказать. Эта конъюнкция целей называется резольвентой (меняется, динамическа) и хранится в виде стека.

Успех достигается тогда, когда резольвента пуста.

Преобразование резольвенты выполняется с помощью преобразования, которое называется редукцией.

Редукцией цели G с пом программы P называется замена цели G телом того правила из P, заголовок которого унифицировался с целью (это правило называется сопоставимым с целью)

Новаяя резольвента выполняется в 2 шага:

- В текущей P выбирается одна из целей и для нее выполняется редукция.

- К полученной конъюнкции целей применяется подстановка, полученная как наибольший общий унификатор цели и заголовка сопоставленного с ним правила.

Если в рез работы алгоритма унификации для редукции был подобран факт, то резольвента уменьшится на одну цель

Пример: $t(X, p(X, Y)) = t(q(W), p(q(a), b))$

	Рег. ячейка		Рабочее поле		Стек
0					$t(X, p(X,Y)) = t(q(W), p(q(a), b))$
1			$t(X, p(X,Y)) = t(q(W), p(q(a), b))$		$X = q(W)$; $p(X,Y) = p(q(a), b)$
2	$X = q(W)$	←	$X = q(W)$	Замена →	$p(X,Y) = p(q(a), b)$
3	$X = q(W)$		$p(q(W), Y) = p(q(a), b)$		$q(W) = q(a)$, $Y = b$
4	$X = q(W)$		$q(W) = q(a)$		$W = a$, $Y = b$
5	$X = q(W)$; $W = a$	←	$W = a$	Замена $W = a$	$Y = b$
6	$X = q(a)$, $W = a$; $Y = b$	←	$Y = b$		Стек пуст

$\theta = \{X = q(a); W = a; Y = b\}$

1.10 Отсчечение

Пролог предусматривает возможность отсечения, которая используется для прерывания поиска с возвратом; отсечение обозначается восклицательным знаком (!). Действует отсечение просто: через него невозможно совершить откат (поиск с возвратом).

Отсечение помещается в программу таким же образом, как и подцель в теле правила. Когда процесс проходит через отсечение, немедленно удовлетворяется обращение к cut и выполняется обращение к очередной подцели (если таковая имеется). Однажды пройдя через отсечение, уже невозможно произвести откат к подцелям, расположенным в обрабатываемом предложении перед отсечением, и также невозможно возвратиться к другим предложениям, определяющим обрабатывающий предикат (предикат, содержащий отсечение).

Существуют два основных случая применения отсечения.

Если вы заранее знаете, что определенные посылки никогда не приведут к осмысленным решениям (поиск решений в этом случае будет лишней тратой времени), - примените отсечение, - программа станет быстрее и экономичнее. Такой прием называют зеленым отсечением. Если отсечения требует сама логика программы для исключения из рассмотрения альтернативных подцелей. Это - красное отсечение.

2 | Лекция 3

Механизм унификации – основной, но не единственный шаг.

Резольвенту использует СИСТЕМА, а не унификация. Первое состояние - вопрос. Док-во истинности заключается в преобразовании резольвенты (уже в другой области).

Стек растущий влево

Система исп доп области

2.1 механизм отката

На каждом новом этапе док-ва истинности вопроса может сложиться одна из 3 ситуаций

- 1) решение (все) найдено – завершение алгоритма

  признаком нового шага является новое состояние резольвенты

- 2) имеется некоторое число возможных дальнейших действий, но неизвестно, какое (какой выбор) приблизит к решениям

- 3) решение не найдено и из данного состояния невозможен переход в новое (тупиковая ситуация) – необходим возврат (откат)

2.2 Дерево поиска решений

Это формализм для исследования возмодных путей решения.

При доказательстве цели G с пом программы P дерево определяется:

- Корень – вопрос,

- вершины – резольвенты (конъюнктивная, в виде стека)

- ребро – переход. На ребре записывается подстановка, соотв определенному выбору. Но в каком порядке – как в бз. Если откат - стрелочка обратно

- лист дерева наз успешной вершиной, если резольвента пуста (надо подписать), и безуспешной – если тупиковая ситуация (нет знания, которое позволяет доказать истинности верхней цели резольвенты)

2.3 Повышение эффективности работы программы на пролог

Предикат отсечения (cut):обозначается !, Отсекает безуспешные ветки поиска решения

Предикат fail – принудительно включает механизм отката. Может только вв конце правила по сути

Названияя – главные функторы (нульместные)

Часто fail пишется сразу после !

r1: - a, b, !, c, d. r2: - ... .

На прямом ходе доказательства ! проницаем(то есть всегда истинный). На обратном ходе – Запрещает использование всех других правил этой процедуры. Будет нет при всех других попытках использовать правило этой процедуры

2.4 списки

Главный функтор – .

[] – пустой список

Голова и хвост

"."(z, "."(b, [])) = [a, b, []] = [a, b]

<имя домена>=<элемент>*

Предикат, позволяющий проверить, является ли аргумент списком.

list(L) :- L=[]. list(L) :- L=[H|T], list(T).

более эффективно list([]). list([_|T]) :- list(T).