18.03.19

PROLOG

мы будем изучать visual prolog

программа на пролог представляет из себя базу знаний и вопрос
в прологе существует единственная синтаксическая конструкция которая называется term (константа, переменная или составной term)
с помощью констант обозначаются некоторые конкретные объекты

как константы могут быть использованы числа и символы

переменные начинаются с заглавных букв
с маленькой - имя/обозначение объекта

переменные могут быть именованными и наименованные
именованные - A1
наименованные - _

переменные используются в программах, чтобы обозначить группу объектов

и переменная и константа обозначают какой-то объект/событие предметной области

функторы - f (функторы являются названием отношения)
f(X, a1)
f1(aB, a1)

term это константы, переменные и составные термины

учится(Иванов, МГТУ)

программа (база знаний) состоит из фактов и правил
факт - безусловная истина, которая фиксируется ….. f(X, a1): T1, T2, …, Tn

у факта нет тела, у него есть функтор и аргумент

с помощью фактов фиксируем знание об объектах

1.04.19

чем больше переменных используется в утверждении базы знаний
тем более в общей форме сформулированном это утверждение

через анонимные переменные нельзя передать данные
их использование необходимо для формализации процесса

именованная переменная уникальна в рамках одного предложения

ПОДСТАНОВКИ И ПРИМЕР ТЕРМА

если факт не содержит переменные то этот факт называется основным

терм в который не входят переменные - основной терм

A(x1, x2, …, xn)

подстановкой называется множество пар вида {xi = ti}, где xi - переменная, ti - терм не содержащий переменные

тэта = {x1 = t1, …, xn = tn}

если A - некоторый терм, а тэта подстановка, которая к нему применена (Атэта), то в результате мы получим пример исходного терма A, т.е. частный случай

применить подстановку - заменить все вхождения переменной xi на ее соответствующие значение ti (B = Атэта)

B - пример
тэта - выражение ….

терм C является общим примеров термов A и B, если существуют такие подстановки тэта1 и тэта2, что C = Атэта1 и C = Втэта2

ЛОГИЧЕСКИЙ ПРОГРАММЫ, ПРОЦЕДУРЫ

знание находится в заголовке правила
правая часть правила содержит условие истинности

с помощью алгоритма кации система подбирает знание

логическая программа - конечное множество правил

процедура- совокупность правил, заголовки которых имеют одно и тоже имя (арность) - количество аргументов

отношение определенное процедурой называется предикатом

если 2 терма, которые имеют одинаковые главные функторы, но разное количество аргументов, то система ….

ДЕКЛОРИРОВАНИЕ ПРОЦЕДУР, СЕМАНТИКА ПРОГРАММЫ

семантикой формы в логике предикатов является процесс связывания формулы с логическим значением

пролог одновременно логический язык и к нему применима императивная семантика программа на прологе может быть интерпретирована с точки зрения процедурной семантики и с точки зрения логической

с позиции декларативной семантики …

с точки зрения процедурной реализации логической программы оказывается важным порядок обработки утверждений базы знаний и порядок обработки (фиксации) условий истинности конкретных правил

УНИФИКАЦИЯ ТЕРМОВ

унификация - операция, которая позволяет формализовать процесс логического вывода

1. двунаправленная передаче параметров процедурам
2. не разрушающая конкретизация переменных
3. проверка условий

термы Т1 и Т2 унифицируются по правилам:
алгоритм унификации после запуска даст ответ: да, можно унифицировать или нет, нельзя

1. если т1 и т2 константы, унифицируются только в том случае, если они совпадают
2. если т1 неконкретизированная переменная или свободная, а т2 константа или составной терм, не содержащий в качестве аргумента т1, то унификация успешна, а переменная т1 конкретизируется …
3. если т1 и т2 неконкретизированные переменные, то их унификация всегда завершается успехом, примем т1 и т2 становятся сцепленными (2мя именами одного и того же (возможно неизвестного) значения), поэтому если одна переменная в дальнейшем получит значение, то вторая в этот же момент получает такое же значение
4. если т1 и т2 составные термы, то они успешно унифицируется если:
    1. у т1 и т2 одинаковые главные функторы
    2. т1 и т2 имеют одинаковые главные арности
    3. успешно унифицируются каждая пара их соответствующих компонентов

АЛГОРИТМ УНИФИКАЦИИ

назначение алгоритма: подобрать подходящее знание

результат работы алгоритма унификации: подстановка (множество значений переменных)

терм S - называется более общим, чем Т, если Т является примером S (Т является частным случаем S), а S не является примером Т

терм S - называется наиболее общим примером 2х термов, если S такой их общий пример, который является более общий по отношению к любому другому их примеру

унификатором 2х термов называется подстановка, которая будучи применена к каждому терму даст одинаковый результат

наиболее общим унификатором 2х термов называется унификатор соответствующий наиболее общему примеру термов

если 2 терма унифицируемым, то существует единственный с точностью до переименования переменных наиболее общий унификатор

САМ АЛГОРИТМ
система запускает алгоритм унификации многократно

стек не единственный
при алгоритме используется рабочая область которая хранит т1=т2
стек для хранения равенств и результирующая ячейка памяти

алгоритм представляет собой цикл
на каждом шаге из стека считывается рабочая область и обрабатывается равество

цикл завершается, если возникает отказ или завершается стек

для работы используется переменная типа флага она имеет значение 0 при успехе или 1 при тупиковой ситуации

начало
занести в стек т1=т2
положить неудача = 0
пока стек не пуст
считать из стека очередное равенство S = T
обработать равенство :
а) если S и T несовпадающие константы то
неудача = 1
выход из цикла
б) S и T одинаковые константы то
следующий шаг цикла, т.е. переход на конец цикла
в) если S переменная, а T терм содержащий S то
неудача = 1
выход из цикла
г) если S переменная, а T терм не содержащий S то
отыскать в стеке и результирующие ячейки все вхождения S и заменить их на T
добавить в результирующую ячейку S = T
следующий шаг цикла
д) если S и T составные термы с разными функторами или арностями то
неудача = 1
выход из цикла
е) если S и T составные термы с одинаковыми функторами или арностями S = f(s1, s2, ..., sn) T = f(t1, t2, ..., tn) то ()
занести в стек равенство s1 = t1, s2 = t2, ..., sn = tn
отчистить рабочее поле
выйти из цикла
если неудача = 0, все успешно, в результирующей ячейке находится подстановка (наиболее общий унификатор)

большие буквы - переменные, маленькие буквы - константы
t(X, p(X, Y)) = t(q(W), p(q(a), b))

шаг	результирующая 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)

шаг	результирующая ячейка	рабочее поле	стек
2	X=q(W)	X=q(W)	p(X, Y)=p(q(a), b) (происходит замена)
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	Y=b
6	X=q(a) W=a	Y=b	пусто
7	X=q(a) W=a Y=b		

ОБЩАЯ СХЕМА СОГЛАСОВАНИЯ ЦЕЛЕВЫХ УТВЕРЖДЕНИЙ
решение задачи с помощью логической программы начинается с задания вопроса G
завершается получением одного из 2х результатов:

1. успешное согласование программы и вопроса (в качестве побочного эффекта будет получена подстановка, которая содержит значение переменных при которых вопрос является примером программы)
2. неудача (возникает в том случае, ЕСЛИ В БАЗЕ ЗНАНИЙ НЕДОСТАТОЧНО ЗНАНИЙ, ЧТОБЫ ОТВЕТИТЬ «Да»)

вычисления с помощью конечной логической программы представляют собой пошаговое преобразование исходного вопроса
на каждом шаге имеется некоторая конъюнкция целей, возможность (выводимость) которой следует доказать
эта конъюнкция целей называется резольвентой

успех достигается тогда, когда резольвента пуста (однократный ответ ДА)
преобразование резольвенты осуществляется при помощи операции - редукции

редукция цели Ж с помощью программы п
назыв замена цели ж телом того правила из тела п заголовок которого унифицируется с целью

такие правила называются сопоставимыми с целью

новая резольвента получается в 2 этапа:

1. в начале в текущей резольвенте выбирается одна из целей для нее выполняется редукция
2. затем к полученной конъюнкции цели применяется подстановка, полученная как наибольший общий унификатор цели и заголовка сопоставленного с ней правила

если было выбрано правило то цель заменилась на тело
если было подобрано знание (заголовок) - заменяет цель на тело

если был факт - тело пустое - редукция будет выполнена, но тело пустое - стек уменьшается на 1 цель (верхнюю в резольвенте)

если резольвента пуста - ответ ДА

в процессе доказательства может возникнуть тупиковая ситуация

ДЕРЕВО ПОИСКА РЕШЕНИЙ
графический образ - который демонстрирует порядок решения

первый вопрос попадает в резольвенту - вершина

алгоритм унификации

новое состояние резольвенты - новая вершина

подстановку нужно записать на дуге

новая резольвента, выбирается новая цель

унификация

и т.д.

если был подобран факт - резольвента меньше

если где-то пустая резольвента - в этом месте ответ да
при этом множество значений переменных тоже выводится

при ответе да
запускается алгоритм отката (на 1 шаг)
и продолжает работать дальше

когда все возможные варианты ответа ДА найдены

под каждым листом дерева нужно писать ДА или ТУПИК

! - системный предикат отсечение
обычно пишется в теле правила

процедура - несколько правил про одно знание

написать поиск максимума в 2х вариантах: с использованием отсечения и без использования

рекурсия:
для доказательства истинности знания нужно сослаться на знание

в теле правила должен быть …

15.04.19

ПРЕДСТАВЛЕНИЕ СПИСКОВ

«.»(a, «.»(b, «.»(c, []))) - от такой формы отказались

[a, b, c] - непустой список
[] - пустой список

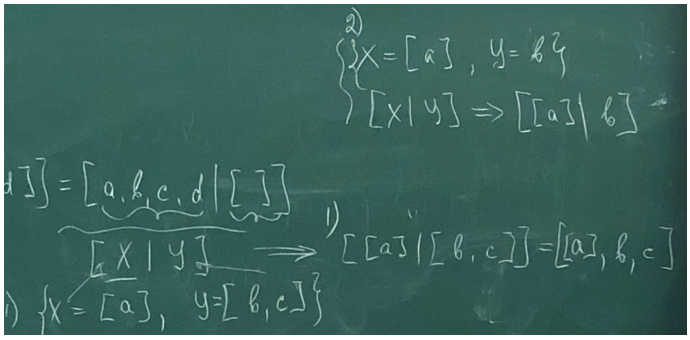
главный функтор - нотация

у списка есть начало и остаток
[a, b | [c,d]] = [a, b, c, d] = [a | [b, c, d]] = [a, b, c | [d]] = [a, b, c, d | [ ]]
остаток обязательно список

именованный переменные могут быть связаны с какими-то значениями

анонимные переменные не связываются с какими-то значениями
они выступают как программные объекты которые позволяют нам выдержать шаблон информации

[X | Y]



1) у нас есть переменная и она расположена в голове списка
[L] = []
[L] = [a | []] => {L = a}
[L] = [a, b, c]
[L, b] = [a, b] => {b = a}
[[L], b] = [a, b]

2) когда переменная в хвосте
[a, L] = [a, b, c]
[a|[L]] = [a|[b, c]] => {L = [b, c]}
— —
[a, L] = [a, b | [c]] => {L = [b, c]}

является ли аргумент списком
list(L):- L = [] list([a, b, c])
list(L):- L = [H | T], list(T).
— — —
list([ ]).
list([_ | T]) :- list(T).

goal
list([a, b, c])

каждый шаг доказательства начинается появлением нового состояния резольвенты
на каждом шаге доказательства выбирается определенная цель
выполняется алгоритм унификации и мы все умираем

правила: list([ ]).
list([_ | T]) :- list(T).

текущая резольвента : list([a, b, c])

шаг 1 ТЦ: list([a, b, c])
ПРI: [ ] = [a, b, c] => унификация невозможна
ПРII: [_ | T1] = [a, b, c] => {T1 = [b, c]}
ТР: list([b, c])
шаг 2 ТЦ: list([b, c])
ПРI: [ ] = [b, c] => унификация невозможна
ПРII: [_ | T2] = [b, c] => {T2 = [c]}
ТР: list([c])
шаг 3 ТЦ: list([c])
ПРI: [ ] = [c] => унификация невозможна
ПРII: [_ | T3] = [c] => {T3 = [ ]}
ТР: list([ ])
шаг 4 ТЦ: list([ ])
ПРI: [ ] = [ ] => успех
ТР - пусто => откат
ТЦ: list([ ])
ПРII: [_ | T4] =[ ] -> no
завершение

принадлежит ли элемент списку
member(X, [H | T]) :- X = H.
member(X, [H | T]) :- member(X, T).

member(X, [X | _]).
member(X, [H | T]) :- member(X, T).

goal
member(a, [b, a, c], R)

заданные
1. посчитать количество элементов списка
2. найти произведение элементов списка
3. объекдинение 2ч списков
удалять элементы из списка и формировать элементы ча четной на нечетной

29.04.19

процесс работы системы системы начинается когда хз когда что-то про правило)

```
append([], L, L).
append([H1 | T1], L2, [H1 | T3]) :- append(T1, L2, T3)


// удалить все вхождения элемента
delete_all(вариант 1: СДОХНУТЬ).
delete_all(_, [ ],[ ]).
delete_all(X, [X | L], L1) :- delete_all(X, L, L1).
delete_all(X, [Y | L], [Y | L1]) :- X <> Y , delete_all(X, L, L1).

          ТР: delete_all(2, [2, 3, 2, 4], R).
шаг1: ТЦ: delete_all(2, [2, 3, 2, 4], R).
          ПРII : {2 = X1, [3, 2, 4] =L1, R = L11}
          ТР: delete_all(2, [3, 2, 4], L11)
шаг2: ТЦ: delete_all(2, [3, 2, 4, L11])
          ПРIII: {2 = X2, 3 = Y2, [2, 4] = L2, L11 = L12}
          ТР: X <> Y или 2<>3
               delete_all(2, [2, 4], L11)
шаг3: ТЦ: 2<>3
              УСПЕХ
          ТЗ: delete_all(2, [2, 4], L11)
шаг4: ТЦ: delete_all(2, [2, 4], L11)
          ПРII: {2 = X4, [4] = L4, L11 = L14}
          ТР: delete_all(2, [4], L14)
шаг5: ТЦ: delete_all(2, [4], L14)
          ПРIII: {2 = X5, 4 = Y5, [ ] = L5, L14 = L15}
          ТР: X<>Y <-> 2<>4
               delete_all(2, [ ], L15)
шаг6: ТЦ: deleye_all(2, [ ], L15)
          ПРI: {L15 = [ ]}
          ТР: пусто, успех  ——> R = [3, 4]
```

ПРЕОБРАЗОВАТЬ СПИСОК К МНОЖЕСТВУ