Решение задач – решение хорошо определенной задачи на основе поискового алгоритма.

Задача является хорошо определенной, если для нее имеется возможность задать пространство возможных решений (состояний), способ просмотра этого пространства с целью поиска конечного (целевого) состояния, соответствующего решенной задачи.

Представление задач в пространстве состояний

Представление, сводящее задачу к подзадачам

Представление задач в виде теорем

Представление задач в пространстве состояний

Задать множества (

S – множество возможных состояний задачи

S0 – множество начальных состояний

F – множество операторов, образующих состояния

G – множество целевых состояний

)

Основы Пролога.

Программа представляет собой декларативное описание отношений некоторой предметной области. Выполнение программы заключается в постановке вопросов, относящихся

Программа представляет собой совокупность утверждений, записанных с использованием предикатов. Предикатом называется логическая функция с произвольным числом аргументов. P(x1,x2,…,xn) – n-местный предикат. P – предикатная буква.

В языке пролог регистрозависимый. В конце утверждения всегда ставиться точка. Есть 2 типа утверждения: факты и правила.

Факт – предикат, являющийся истинным утверждением.

Символьные константы начинаются со строчной буквы или в кавычках. Переменные начинаются с прописной буквы. Правило – утверждение, которое истинно при некоторых условиях. Правило состоит из заголовка, (головы) и тела. :- соединитель, соответствует импликации. «,» - и, «;» - или.

«%» - однострочный комментарий

«/\*…\*/» - многострочный комментарий

Совокупность правил и фактов образуют базу данных.

Целевое утверждение – «?-утверждение».

Обращение инициирует процесс поиска. Система начинает искать факты, сопоставимые с этим запросом. Два факта сопоставимы, если совпадают имена предикатов, и соответствующие аргументы попарно сопоставимы.

В общем виде правила формируются как P:-P1,P2,…,Pn.

Где Р1,… - атомарные элементарные формулы.

Атомарная формула имеет вид P(t1,t2,…,tn), где t1, … - термы.

Терм может быть константой, переменной или составным термом. Атомы языка представляют символьные константы.

Областью действия переменных является утверждение.

Переменные, которым присвоено значение, называются конкретизированными. Существуют так же анонимные переменные – «\_» – переменная с любым значением.

Структура (составной терм) – объект, состоящий из нескольких компонент: атом (функтор) и последовательность термов (компоненты структуры). Компоненты структуры разделяются запятыми. Число компоненты в структуре – её арность.

Одной из основных операций над термом – сопоставление. Правило сопоставления:

* X, y константы и их значения равны
* Если х представлен константой или структурой, а терм у – не конкретизированной переменной, то термы х и у сопоставимы и у принимает значение х.
* Если термы х и у структуры, то они сопоставимы, если сопоставимы все внутренние термы и функторы.

При рассмотрении программы рассматривается 2 смысла:

1. Декларативный связан с отношениями, объявленными в программе, он представляет собой достижимо ли целевое утверждение.
2. Процедурный, определяет, как система обрабатывает отношения

В прологе есть ка обработка символьных констант так и арифметических операций и операторов сравнений. Оператор is заставляет выполнять арифметические операции.

Списки и рекурсия. Управление возвратом. Отрицание. Метаусловия.

Список – структура данных, составленная из произвольного числа однотипных элементов, отделенных запятой, заключенная в квадратные скобки.

Добавление элемента в список доб(Х,У,[Х|У]).

Рекурсивными называются предикаты, которые содержат ссылки сами на себя.

Вхождение элемента member(H,[X|T]). Элемент H содержится в [X|T]

Соединение списков append(X,Y,Z)

Удаление

Организация циклов.

Состояние искусственной неудачи – fail. Fail осуществляет возврат к следующей альтернативе.

Предикаты называются недетерминированные.

Цикл с заданным числом повторений

Repeat(1):-!.

Repeat(N):-true; M is N-1, repeat(M).

Read считывает из потока очередной терм и сопоставляет его.

Предикаты декомпозиции термов name(A, L) истина, если L последовательность ASCII кодов представляющих терм A.

Есть терм F с арностью А

CSP – задачи

Целью является нахождение переменных, удовлетворяющих определённым ограничениям.

Решаются задачи

* - назначение ресурсов
* - календарное планирование
* - обработка изображений
* - автоматическая генерация тестов
* - понимания ЕЯ
* - конъюнктивные запросы к БД
* - комбинаторная оптимизация

Задача поиска в пространстве состояний – набор черных ящиков (состояний)

Состояния и проверка целей соответствуют стандартному представлению.

Любая задача CSP определяется совокупностью трех составляющих:

1) множеством переменных X1, X2,…,Xn;

2) областью определения каждой переменной D1, D2,…,Dn;

3) множеством ограничений (отношений) С1, С2,…,Сm, каждое из которых включает некоторое подмножество переменных и задает допустимые комбинации значений для этого подмножества.

*Состояние* задачи определяется путем присваивания значений некоторым или всем переменным. Присваивание, которое не нарушает никаких ограничений, называется *совместимым*. *Полным* называется присваивание, в котором участвует каждая переменная, а решением задачи CSP является полное присваивание, которое удовлетворяет всем ограничениям.

Задачу CSP удобно представлять в виде графа ограничений, узлы которого представляют переменные задачи, а дуги - ограничения.

Переменные задачи могут быть дискретными или непрерывными, с конечными или бесконечными областями определений. Ниже рассматриваются только простые CSP задачи с дискретными переменными, характеризуемыми конечной областью определения. Если максимальный размер области определения равен d, то в худшем случае количество возможных полных присваиваний оценивается величиной O(dn). Поэтому время решения задачи экспоненциально зависит от количества переменных. Однако в большинстве практических случаев алгоритмы CSP общего назначения позволяют решать задачи на несколько порядков более крупные, например, по сравнению алгоритмами поиска решений в пространстве состояний.

Ограничения (отношения) могут быть унарными, бинарными и ограничениями высокого порядка в зависимости от количества переменных. Унарные ограничения ограничивают значение одной переменной. Унарное ограничение можно устранить, удалив соответствующее значение из области определения переменной, нарушающее это ограничение. Бинарное ограничение связывает между собой две переменные. CSP задача, в которой используются только бинарные ограничения, может быть представлена графом ограничений. В ограничениях высокого порядка участвуют три и больше переменных. Ограничение высокого порядка может сведено к бинарным ограничениям путем введения вспомогательных переменных .

Рассмотрим 4 метода решения CSP задач: «генерируй и тестируй», поиск с возвратами, поиск с предварительной проверкой, поиск с распространением ограничения

1. Метод «генерируй и тестируй» (generate and test). Метод также называют методом «образуй и проверь» [2,11]. В соответствии с этим методом генерируются все возможные полные присваивания переменным, и каждое из них тестируется (проверяется) на совместимость. Структура соответствующей программы весьма проста и выглядит в виде вложенных циклов по каждой переменной. В самом внутреннем цикле выполняется проверка каждого ограничения. Если они все выполняются, то текущее полное присвоение — решение задачи.
2. В большинстве случае «слепая» генерация всех полных присваиваний весьма неэффективна, т.к. приводит к разрастанию дерева поиска.Поиск с возвратами (backtracking search). По сути, это тот же метод «генерируй и тестируй», но организованный в виде поиска в глубину, в котором присваивается значение очередной переменной, проверяются ограничения и выполняется возврат, если присвоение не допустимо. Таким образом, проверка ограничений как бы погружается в процесс генерации решения, что позволяет ограничить разрастание дерева поиска. Эффективность метода зависит от порядка выбора переменных. Наиболее часто для определения порядка выбора переменных используют две эвристики: степенную эвристику, MRV-эвристику (MRV ― Minimum Remaining Values) и наименее ограничительное значение. Степенная эвристика позволяет уменьшить степень ветвления за счет выбора переменной, которая участвует в наибольшем количестве ограничений. MRV-эвристика предусматривает выбор переменной с наименьшим количеством оставшихся допустимых значений. После выбора одной переменной необходимо принять решение в каком порядке должны просматриваться значения, позволяя сохранить гибкость максимальную гибкость для последующих присваиваний. Такая переменная с наибольшей вероятностью, вскоре приведет к неудаче, усекая тем самым дерево поиска. Степенная эвристика обычно используется для начального выбора переменных, а MRV-эвристика в ходе дальнейших присваиваний. В процессе поиска с возвратами множество переменных, которым уже присвоены значения, на каждом шаге расширяется на очередную переменную, если выполняются соответствующие ограничения. Процесс завершается, если получено полное присваивание.
3. Метод предварительной (опережающей) проверки (forward checking). В случае поиска с возвратами ограничения, в которых участвует некоторая переменная, учитываются непосредственно в момент, когда происходит назначение значения этой переменной. Но выполняя опережающую проверку некоторых ограничений на предшествующих этапах поиска, можно резко сократить пространство поиска. В соответствии с методом предварительной проверки в момент присваивания значения переменной X просматривается каждая переменная Y, которой не присвоены значения и которая связана с X некоторым ограничением. При этом из области определения Y удаляется любое значение, которое несовместимо со значением, присвоенным переменной X.
4. Метод распространения ограничения. Предварительная проверка не позволяет обнаруживать все несовместимости. Распространение ограничения — это общее название методов обнаружения потенциальных несовместимостей на ранних этапах решения задачи за счет распространения последствий применения некоторого ограничения к одной из переменных. Для быстрого распространения ограничений выполняется проверка совместимости дуг. Дуга (X, Y ) называется совместимой, если для каждого значения х из области определения переменной X существует некоторое значение y из области определения переменной Y, которое удовлетворяет бинарному ограничению между этими переменными. Обратим внимание на то, что понятие совместимости дуг является направленным, т.е. если дуга (X, Y ) совместима, то это не означает, что обратная дуга (Y, X ) также совместима. Проверку совместимости дуг можно использовать либо в качестве этапа предварительной обработки перед началом процесса поиска, либо в качестве этапа распространения ограничения после каждого присваивания во время поиска. Проверка совместимости дуг требует дополнительных затрат времени, которые в наихудшем случае составляют O(n2 d3). По этой причине в большинстве случаев при решении простых CSP задач ограничиваются применением поиска с возвратами или поиска с предварительной проверкой.