Лекция № 12

Методы поиска. Часть 2

Search methods. Part 2

Поиск методом редукции

При поиске *методом редукции* **решение задачи** сводится к **решению** совокупности образующих **ее подзадач**.

Этот процесс повторяется для каждой подзадачи до тех пор, пока каждая задача из полученного набора подзадач, образующих решение исходной задачи, не будет иметь *очевидное решение*.

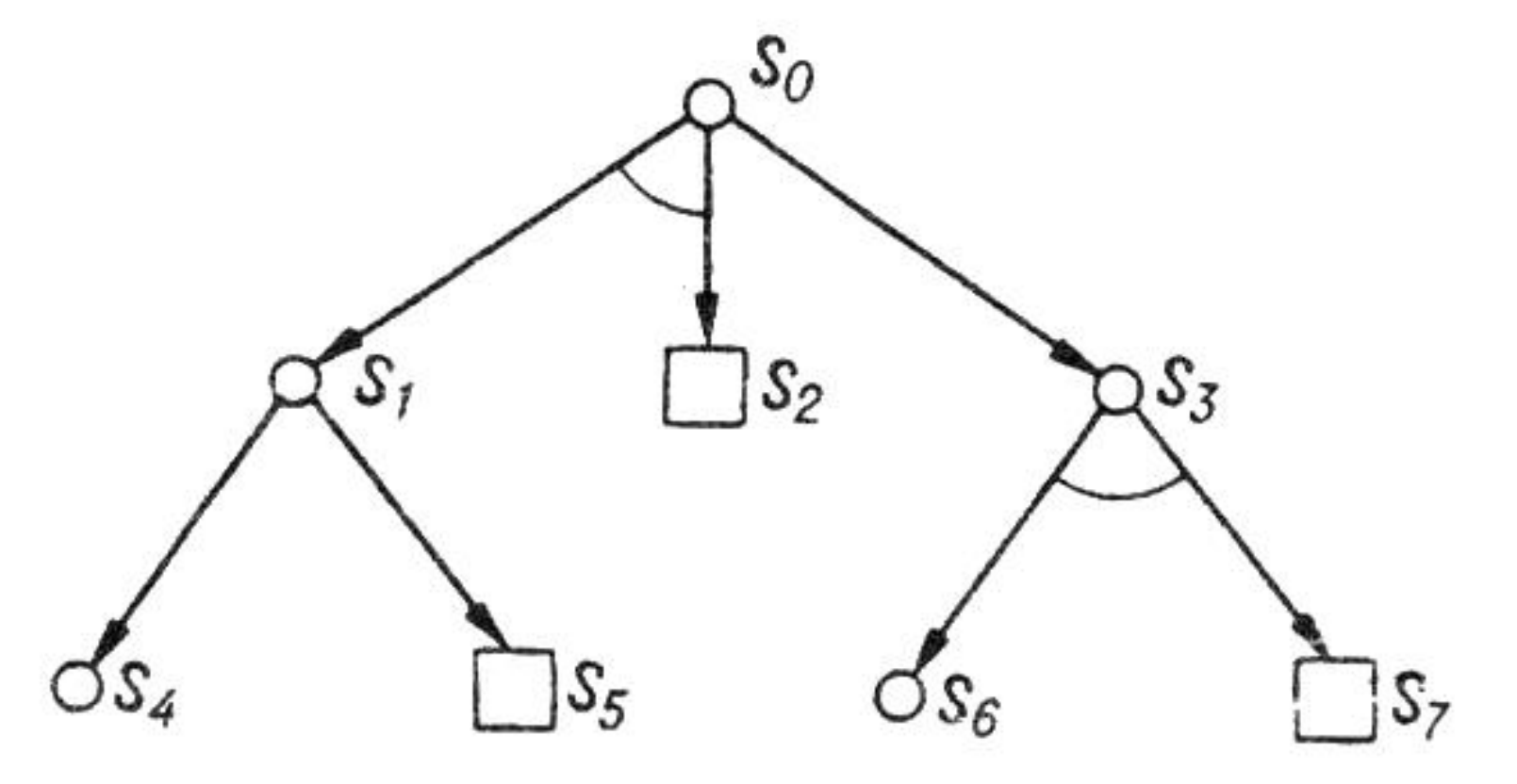
Подзадача считается **очевидной**, если ее решение общеизвестно или получено ранее.

Процесс решения задачи разбиением ее на подзадачи можно представить в виде специального направленного графа **G**, называемого **И/ИЛИ графом**.

Каждой вершине этого графа ставится в соответствие описание некоторой задачи (подзадачи).

Поиск методом редукции

В графе выделяют два типа вершин: *конъюнктивные* вершины и *дизъюнктивные* вершины.



Поиск методом редукции

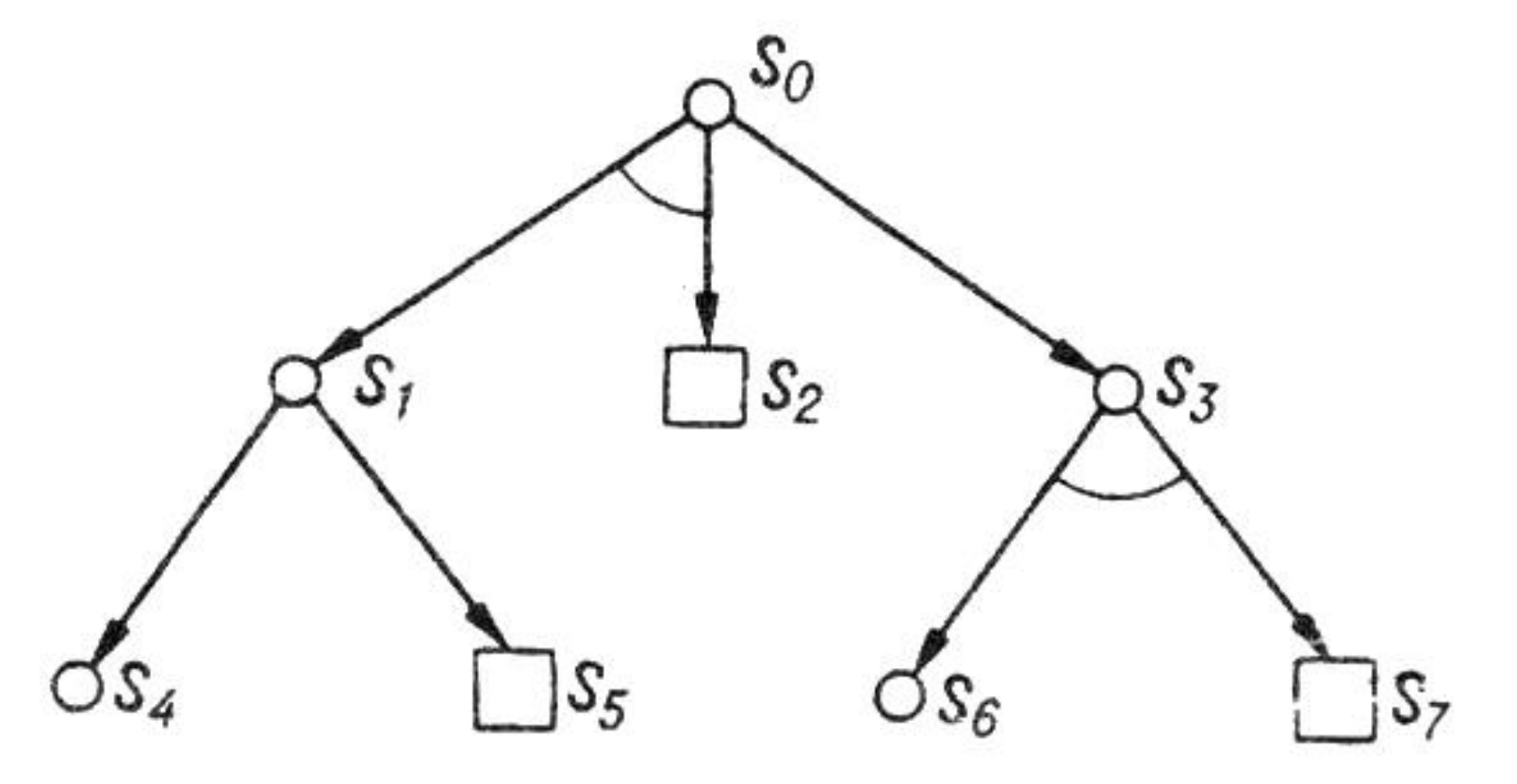
Конъюнктивные вершины (вершины типа "И“) вместе со своими дочерними вершинами интерпретируются так: решение задачи сводится к решению **всех** ее подзадач, соответствующих дочерним вершинам конъюнктивной вершины.

Дизъюнктивные вершины (вершины типа "ИЛИ“) вместе со своими дочерними вершинами интерпретируются так: решение задачи сводится к решению **любой из** ее подзадач, соответствующих дочерним вершинам дизъюнктивной вершины.

В множестве вершин И/ИЛИ графа выделяют подмножество *начальных* вершин, т.е. задач, которые следует решить, и подмножество *конечных* (*целевых*) вершин, т.е. заведомо разрешимых задач.

Поиск методом редукции

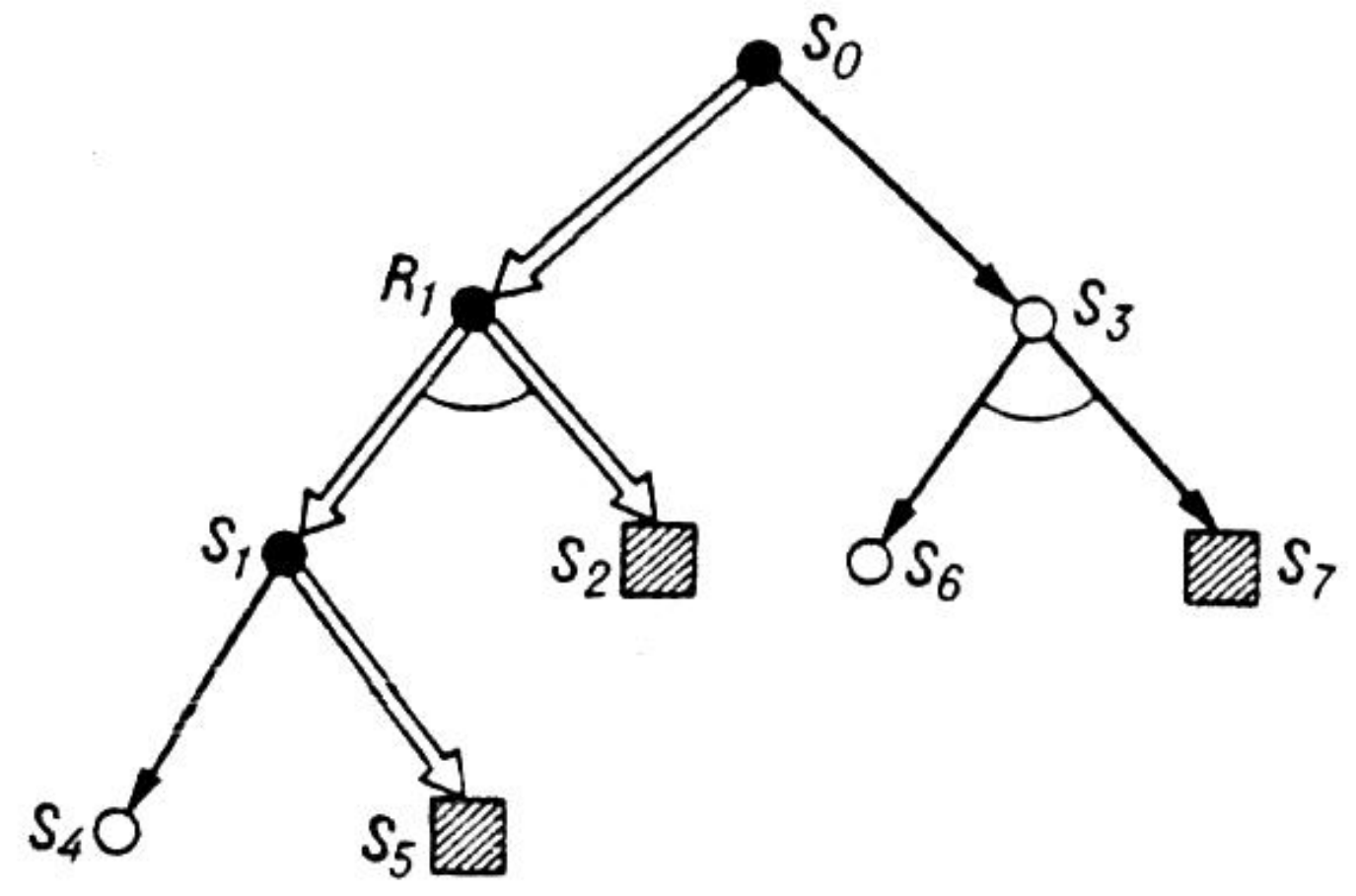
Графическое представление процесса разбиения задачи на подзадачи.



Решение задач S2,S5 и S7 предполагается известным, решение задач S4 и S6 неизвестно.

Поиск методом редукции

Пример И/ИЛИ графа



В связи с тем, что в И/ИЛИ графе каждая вершина относится только к одному типу (либо И, либо ИЛИ), то для записи графа, изображенного выше была введена дополнительная вершина (R1).

Поиск методом редукции

Решение задачи при поиске методом редукции сводится к нахождению в И/ИЛИ графе ***решающего графа***.

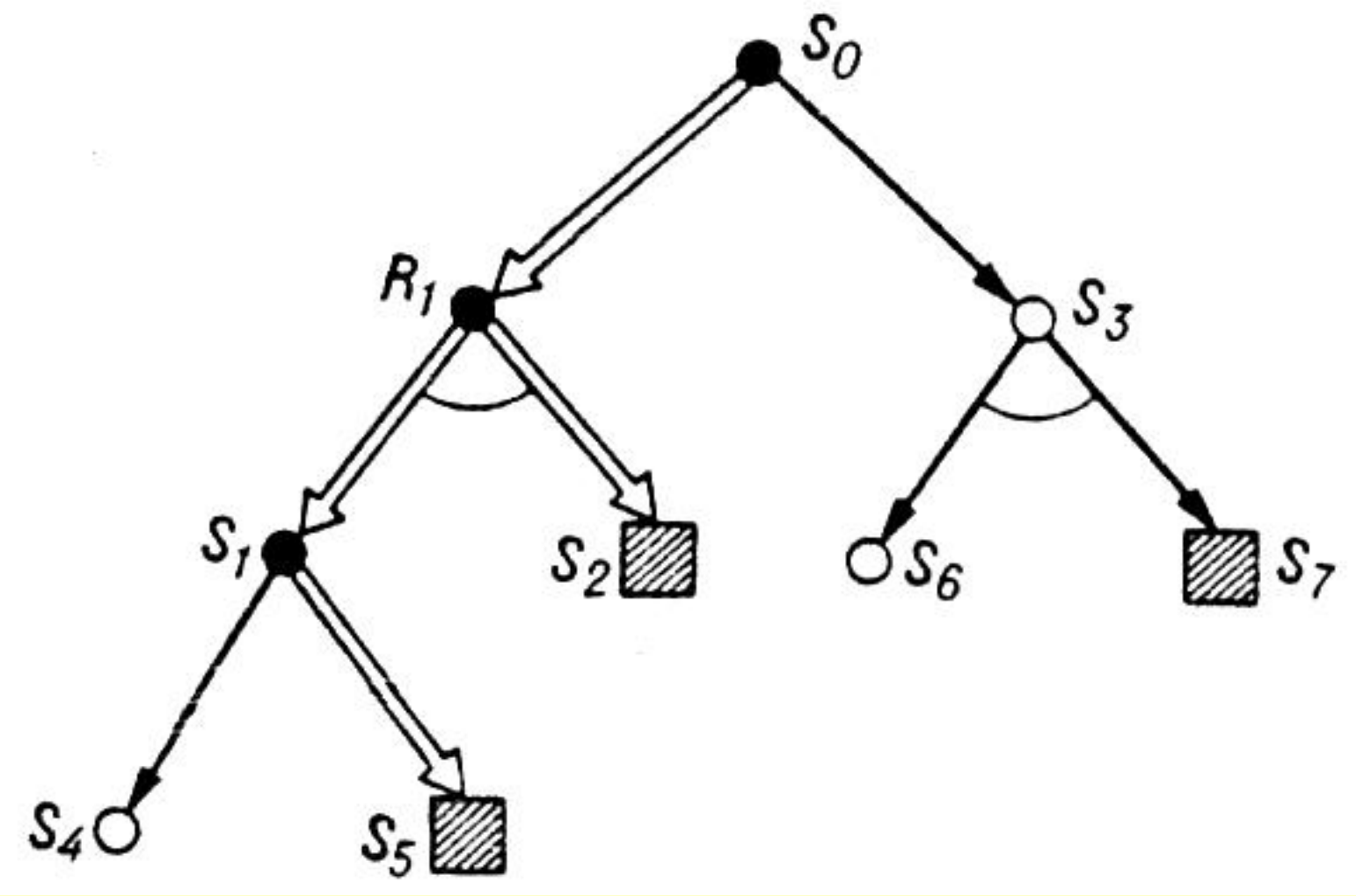
Цель процесса поиска в И/ИЛИ графе — показать, что начальная вершина ***разрешима***, т.е. для этой вершины существует *решающий граф*.

Определение разрешимой вершины в И/ИЛИ графе можно сформулировать рекурсивно следующим образом:

1. Конечные (целевые) вершины разрешимы, так как их решение известно по исходному предположению.
2. Вершина ИЛИ разрешима тогда и только тогда, когда разрешима по крайней мере одна из ее дочерних вершин.
3. Вершина И разрешима тогда и только тогда, когда разрешима каждая из ее дочерних вершин.

Таким образом, ***решающий граф*** определяется как подграф из разрешимых вершин, показывающий, что начальная вершина разрешима.

Поиск методом редукции



Решающий граф включает следующие вершины: S0, R1 S2, S1, S5.

Поиск методом редукции

Для некоторой подзадачи может быть неизвестно ни ее решение, ни способ сведения ее к более простым подзадачам. Такая подзадача называется **неразрешимой**. Определение *неразрешимой вершины* в И/ИЛИ графе можно сформулировать аналогично определению разрешимой.

Заметим, что метод сведения задач к подзадачам является в некотором роде обобщением подхода с использованием пространства состояний. Действительно, перебор в пространстве состояний можно рассматривать как тривиальный случай сведения задачи всегда к одной подзадаче.

Для И/ИЛИ графа, так же как для поиска в пространстве состояний, можно определить поиск в глубину и поиск в ширину как в прямом, так и в обратном направлении.

Поиск методом «генерация – проверка»

Процесс поиска может быть сформулирован в терминах **"генерация — проверка".**

Действительно, пространство поиска (пространство состояний или И/ИЛИ граф), как правило, явно не задано. Поэтому для осуществления процесса поиска необходимо генерировать очередное возможное решение (состояние или подзадачу) и проверить, не является ли оно результирующим.

Разумно потребовать, чтобы генератор удовлетворял требованиям *полноты* и *неизбыточности*.

Говорят, что генератор является **полным**, если он обеспечивает генерацию всех возможных решений.

Генератор является **неизбыточным**, если он генерирует каждое решение только один раз.

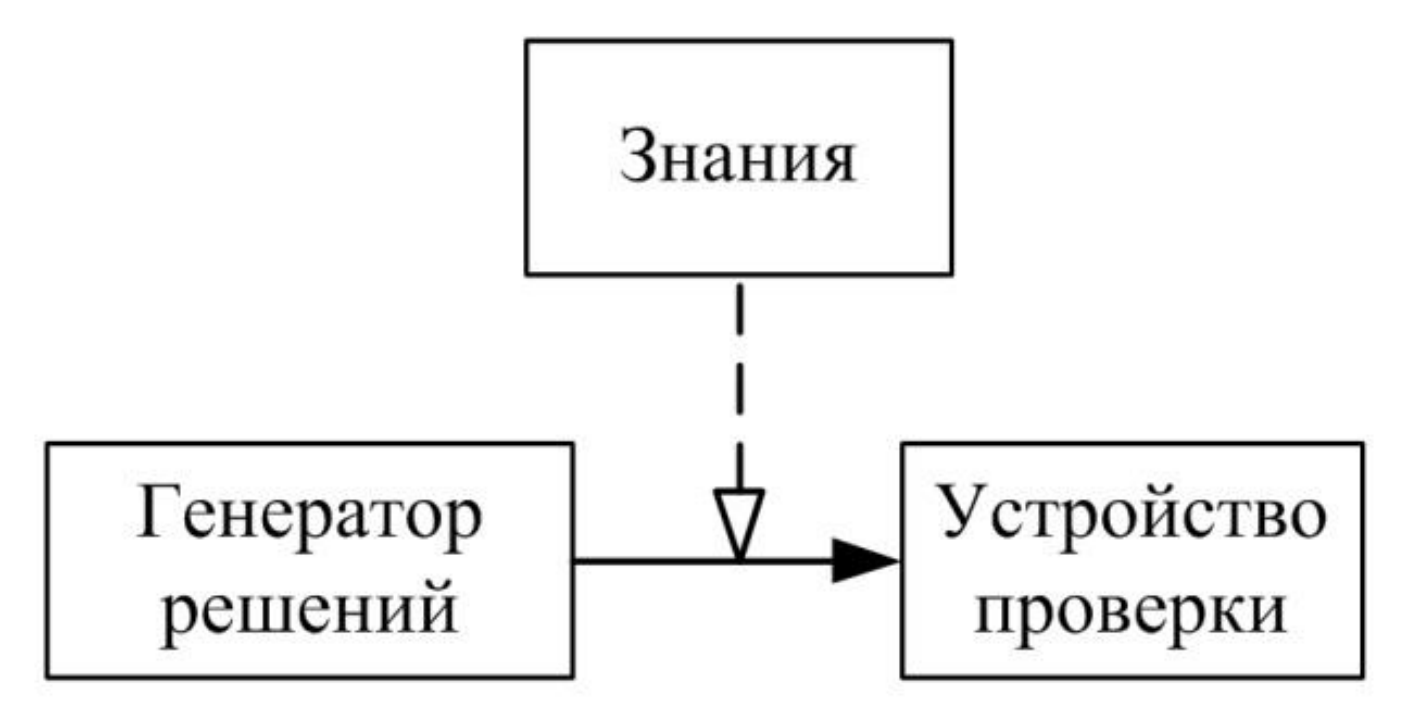
Поиск методом «генерация – проверка»

Обеспечение свойства неизбыточности является важным, но трудновыполнимым, так как в соответствии с этим требованием не допускается генерация не только тождественных, но и синонимичных решений.

Например, если задача генератора — синтезировать все фразы русского языка, то весьма трудно (если вообще возможно) сделать такой генератор неизбыточным.

Поиск методом «генерация – проверка»

При генерации текущего возможного решения (состояния или подзадачи) возникает *проблема распределения знаний* между **генератором** и **устройством проверки**.



Поиск методом «генерация – проверка»

При слепом и эвристическом поиске генератор имеет минимальные знания о проблемной области, достаточные для генерации всех возможных решений, а устройство проверки определяет, не является ли очередное решение целевым.

В принципе некоторые знания можно перенести из устройства проверки в генератор, чтобы он не генерировал решения, которые заведомо не могут привести к успеху.

По сути дела, увеличение знаний генератора о проблемной области приводит к сокращению пространства, в котором осуществляется поиск. Однако при этом повышаются затраты на генерацию каждого очередного состояния (подзадачи).

Поиск методом «генерация – проверка»

Выделяется важный вид метода «генерация — проверка»:

«иерархическая генерация — проверка».

В этом случае на верхнем уровне генератор вырабатывает не полное, а *частично определенное решение* (кратко - частичное решение).

Каждое **частичное решение** описывает, например, не всё состояние, а только его некоторую часть, определяя таким образом класс возможных состояний.

Идея состоит в том, что устройство проверки может уже по виду частичного решения определить, что оно (а следовательно, и все полные решения, которые могут быть получены из него) не ведет к успеху.

Если же проверка не отвергает частичное решение, то на следующем уровне генератор вырабатывает из данного частичного решения все полные решения, а устройство проверки определяет, являются ли они целевыми.

Поиск в иерархии пространств

Методы поиска в одном пространстве не позволяют решать сложные задачи, так как с увеличением размера пространства время поиска экспоненциально растет.

При большом размере пространства поиска имеет смысл разбить всё пространство на отдельные **подпространства** и осуществлять поиск сначала в каждом из них.
Тогда можно сказать, что пространство поиска представлено **иерархией пространств**.

«Введение островков планирования уменьшает время поиска по экспоненте»
(М. Минский, 1963, 1964 гг.).

Поиск в иерархии пространств

Идею Минского об иерархии пространств можно развить, допустив в иерархии не только конкретные, но и **абстрактные пространства**, т.е. пространства которые имеют описание только наиболее важных сущностей.

Использование таких неполных описаний позволяет существенно сократить пространство и ускорить решение задачи.

Поиск в иерархии пространств

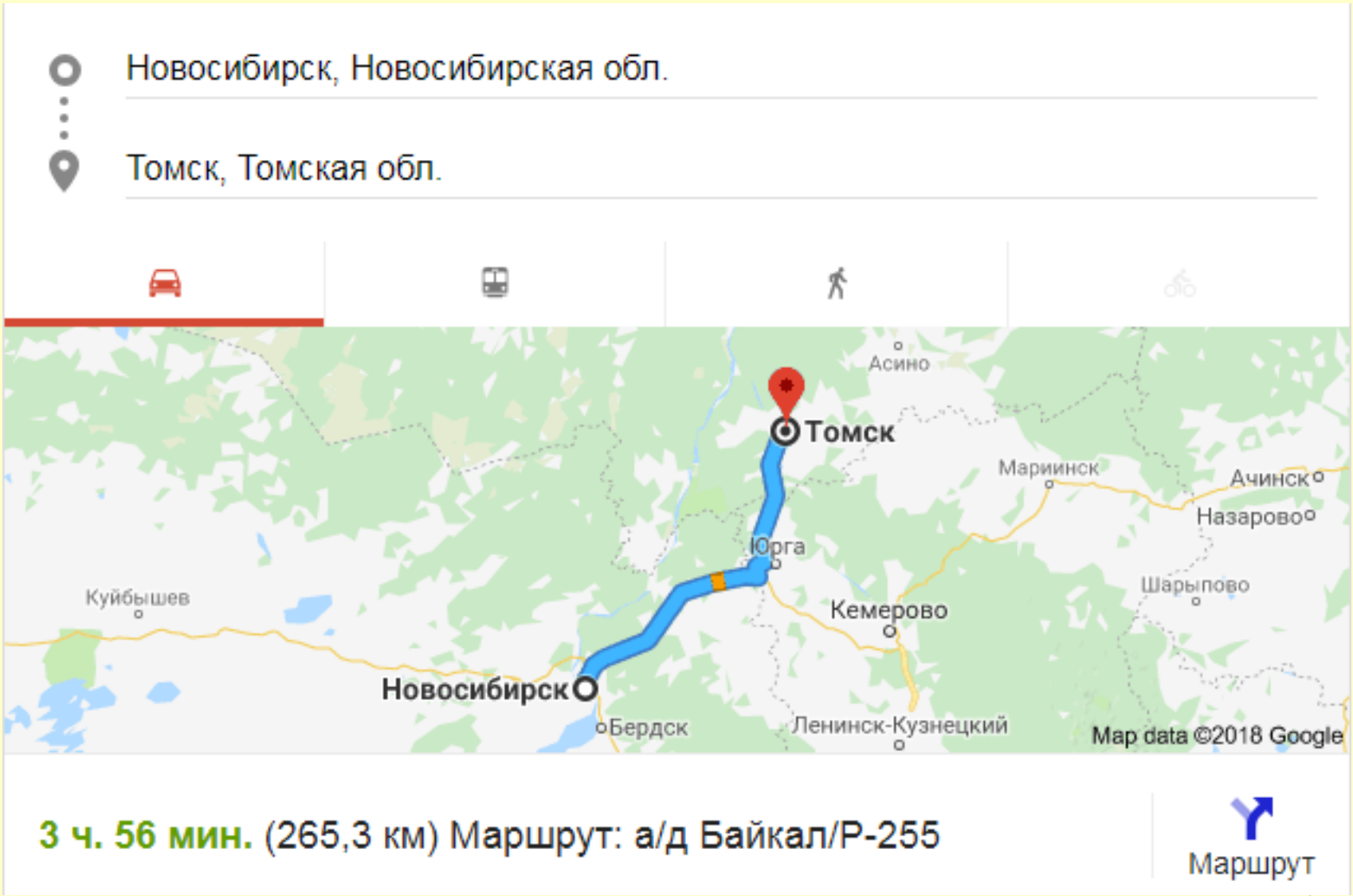
Классический пример использования абстрактных пространств − **задача определения кратчайшего пути на карте**.

Пусть требуется переехать из центра города ***A*** (Новосибирск) в центр города ***B*** (Томск).

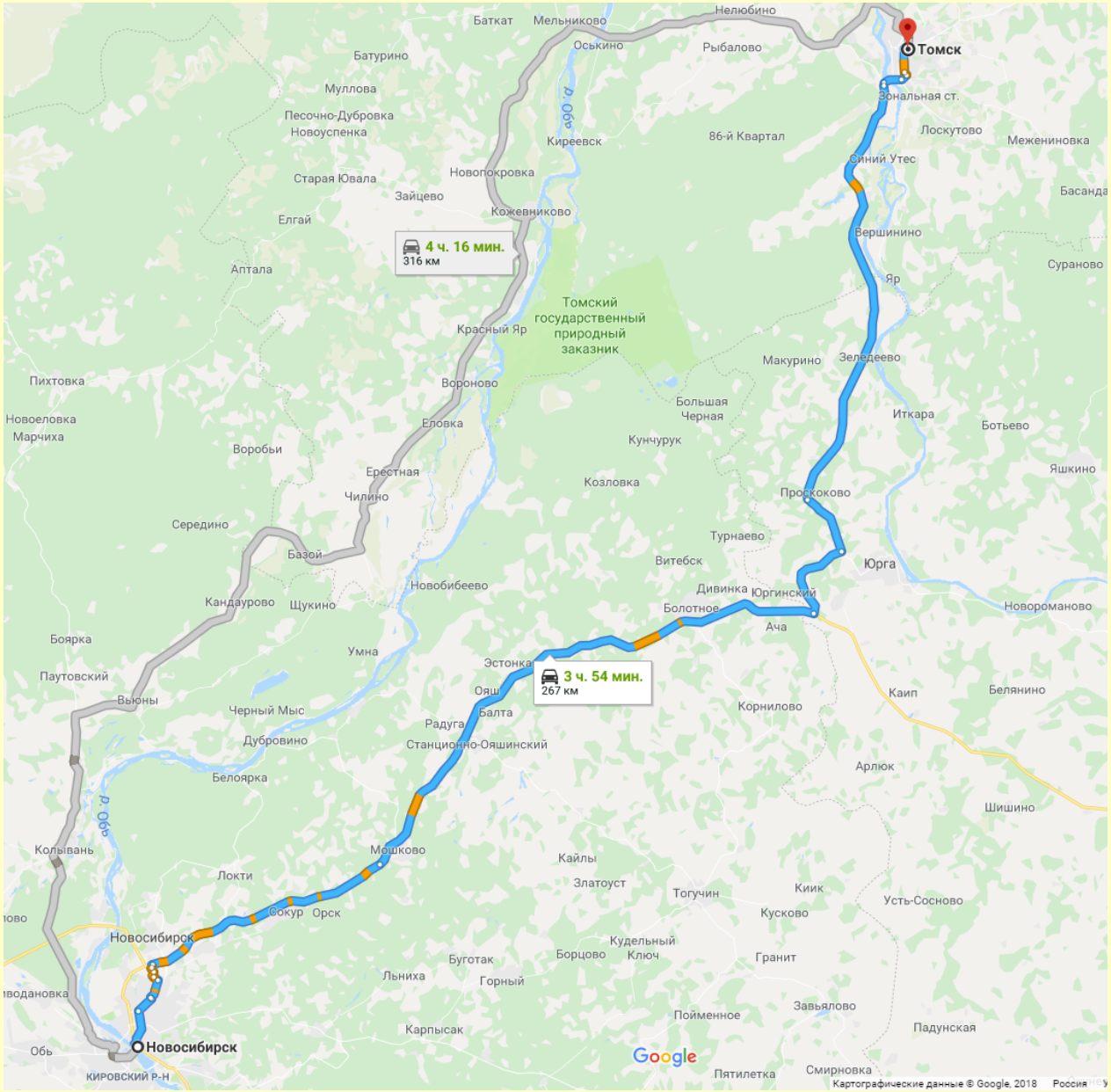
Если осуществлять поиск требуемого пути на детальной карте, содержащей все улицы во всех городах, встретившихся по дороге, то задача может стать практически неразрешимой.

Поэтому при определении пути из города ***A*** в город ***B*** целесообразно сначала спланировать маршрут по крупномасштабной карте (т.е. осуществить поиск в абстрактном пространстве), а затем по детальной карте спланировать выезд из города ***A*** и въезд в город ***B***.

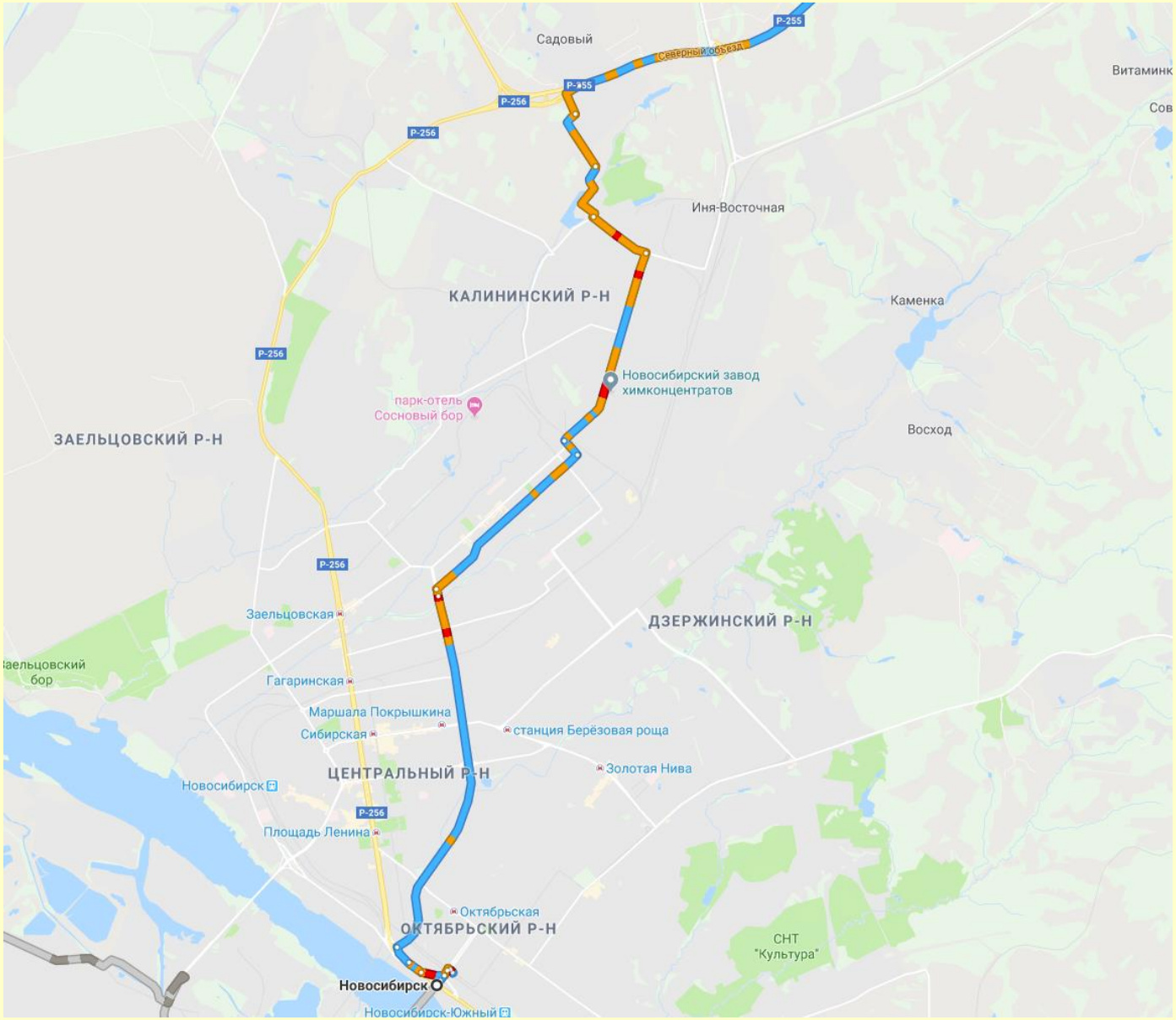
Поиск в иерархии пространств



Поиск в иерархии пространств



Поиск в иерархии пространств



Поиск в иерархии пространств

Далее рассматриваются методы, использующие общую идею иерархии пространств, но отличающиеся предположениями о природе этих пространств.

Большей частью эти методы опираются на абстрактные пространства.

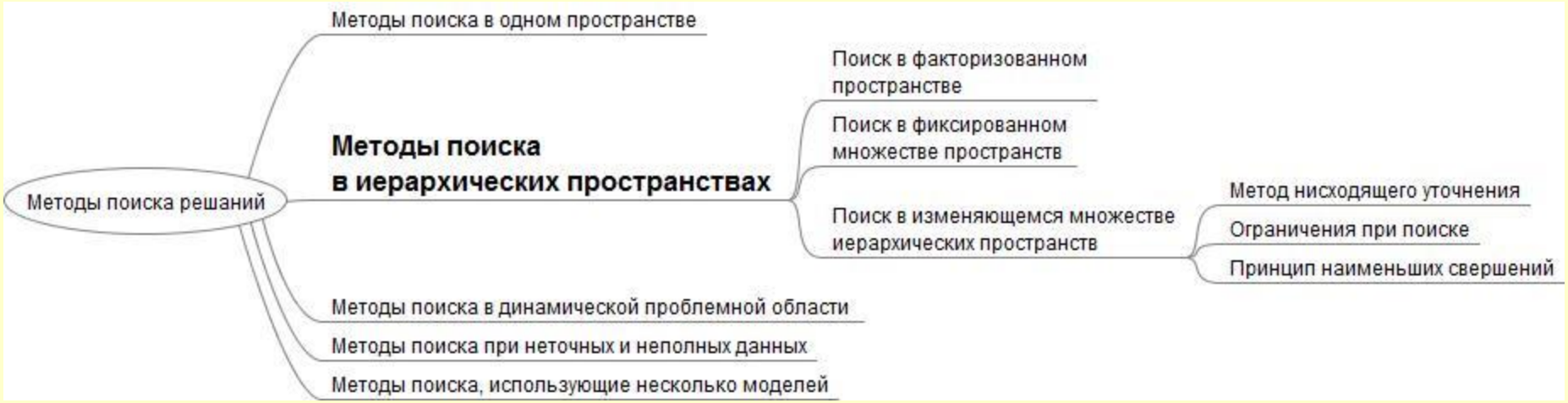
При этом **абстракция** должна

– подчеркнуть важные особенности рассматриваемой задачи,

– позволить разбить задачу на более простые подзадачи и

– определить последовательность подзадач (план решения), приводящую к решению основной задачи.

Поиск в иерархии пространств



1. Поиск в факторизованном пространстве

Во многих приложениях требуется найти все решения.

Примерами таких областей являются интерпретация данных, постановка диагноза и др.

Действительно, в случае постановки диагноза нас интересуют все, а не отдельные болезни пациента.

Однако пространство поиска в практических приложениях бывает столь велико, что не позволяет применить слепые методы поиска. Применение эвристических методов в данном случае, как правило, также исключено, так как они не обеспечивают получение всех возможных решений.

1. Поиск в факторизованном пространстве

Если пространство поиска удается ***факторизовать***, то поиск даже в очень большом пространстве можно организовать достаточно эффективно.

Пространство называется **факторизованным**, если оно разбивается на непересекающиеся подпространства (классы) частичными (неполными) решениями;
причем по виду частичного решения можно определить, что оно не приведет к успеху, т.е. что все полные решения, образованные из него, не приведут к целевым решениям.

1. Поиск в факторизованном пространстве

Поиск в факторизованном пространстве осуществляется на основе метода ***иерархическая "генерация—проверка"***.

Генератор вырабатывает текущее частичное решение, затем проверяется, может ли это решение привести к успеху.

Если это решение отвергается, то из рассмотрения устраняются все полные решения этого класса.

Если текущее частичное решение не отвергается, то генератор вырабатывает на его основе все полные решения, а устройство проверки определяет, являются ли эти решения целевыми.

2. Поиск в фиксированном множестве пространств

Применение рассмотренного выше *метода факторизации пространства* ограничено тем, что для ряда областей не удается по частичному решению сделать заключение о его непригодности.

Примерами таких областей являются задачи планирования и конструирования. Действительно, как правило, по фрагменту плана или конструкции нельзя сказать, что этот фрагмент не может являться частью полного решения.

В простейшем случае пространство поиска разбивается на **фиксированную последовательность подзадач** (подпространств), с помощью которых можно решить любую входную задачу.

2. Поиск в фиксированном множестве пространств

Подобный метод использован в экспертной системе R1, которая на основании заказа покупателя на требуемую ему конфигурацию системы VAX выдает диаграммы, изображающие пространственные взаимосвязи компонент VAX, и указывает какие компоненты необходимы для функционирования заказанной конфигурации.

Система R1 разбивает общую задачу на шесть подзадач. Порядок, в котором вызываются эти задачи, зависит от заказанной конфигурации. Действия, выполняемые каждой подзадачей, зависят от комбинации заказанных компонент и способа их взаимосвязи. В системе каждой подзадаче соответствует свой набор правил, т.е. каждая подзадача решается в своем подпространстве.

3. Поиск в изменяющемся множестве иерархических пространств.

Метод нисходящего уточнения

В ряде приложений не удается все решаемые задачи свести к фиксированному набору подзадач.

Примерами таких приложений являются **задачи планирования перемещений (объектов) в пространстве**.

План решения задачи в данном случае должен иметь переменную структуру и не может быть сведен к фиксированному набору подзадач.

Для решения подобных задач может быть использован ***метод нисходящего уточнения*** *(top-down refinement).*

3. Метод нисходящего уточнения

Для того чтобы упростить процесс решения некоторой задачи в сложном пространстве, целесообразно получить обобщенное (более простое) пространство и попробовать получить решение в этом пространстве. Указанный прием можно повторять многократно.

Таким образом мы можем получить последовательность пространств, упорядоченных по степени абстрактности:

$$К \rightarrow A^1 \rightarrow A^2 \rightarrow \ldots \rightarrow A^n$$

Формирование следующего более абстрактного пространства осуществляется путем игнорирования части описаний менее абстрактного пространства (на первом шаге — конкретного пространства).

3. Метод нисходящего уточнения

Игнорирование описаний осуществляется на основе ранжирования описаний по степени важности.
Часто ранжирование осуществляется на основе учета степени неизменности фактов (наиболее абстрактны те описания, которые не могут изменяться).

При этом абстрактные пространства, с одной стороны, должны для упрощения решения задачи обеспечивать значительное упрощение исходного пространства, а с другой стороны, должны быть подобны друг другу и конкретному пространству, чтобы процесс "нисходящего" переноса решения из более абстрактных пространств в менее абстрактные не требовал больших вычислительных затрат.

3. Метод нисходящего уточнения

При этом полный процесс решения задачи можно представить как "нисходящее" движение в иерархии пространств от наиболее абстрактного к конкретному, в котором получается окончательное решение:

$$A^n \rightarrow A^{n-1} \rightarrow \ldots \rightarrow A^1 \rightarrow К$$

Существенной характеристикой такого процесса является поиск решения задачи в текущем абстрактном пространстве, затем преобразование этого решения в решение более низкого уровня и т.д.

Причем на каждом уровне вырабатывается окончательное решение и только затем осуществляется переход на следующий, более конкретный уровень.

Внутри каждого уровня подзадачи рассматриваются как независимые, что создает частичное упорядочение абстрактных состояний.

3. Метод нисходящего уточнения

Метод нисходящего уточнения **базируется** на следующих предположениях:

1) возможно осуществить частичное упорядочение понятий области, приемлемое для всех решаемых задач;

2) решения, принимаемые на верхних уровнях, нет необходимости отменять на более нижних.

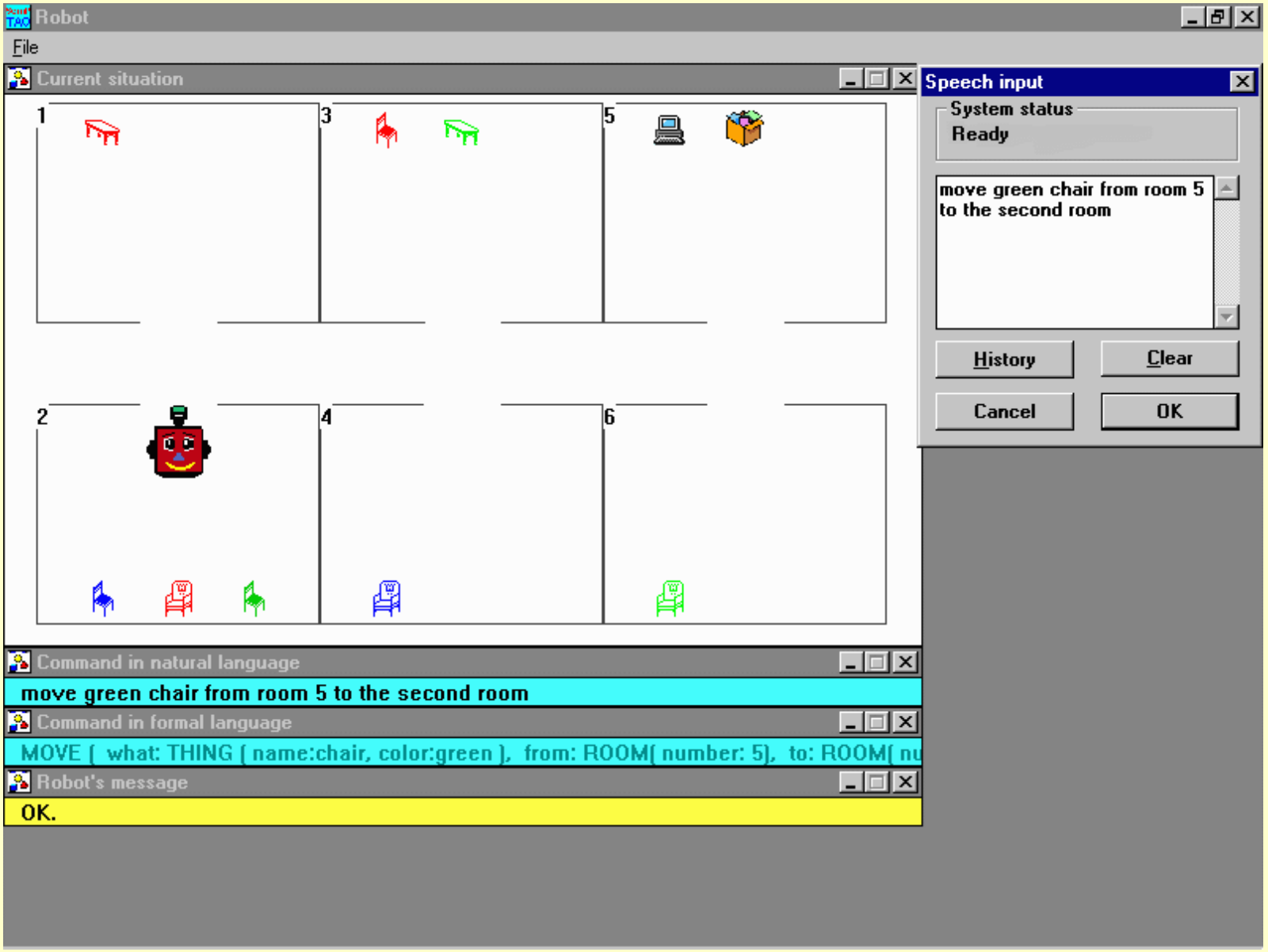
3. Метод нисходящего уточнения. Пример

Наиболее известным примером использования данного метода является **система ABSTRIPS**. Она составляет план перемещения роботом объектов (ящиков) между комнатами.

Робот действует в мире, содержащем описание комнат, расположение дверей в комнатах, состояние дверей (открыты, закрыты), местонахождение объектов, местонахождение робота.

Робот умеет выполнять ряд действий: перемещаться по комнате, переходить из одной комнаты в другую, открывать дверь, толкать объекты и т.п.

3. Метод нисходящего уточнения. Пример



3. Метод нисходящего уточнения. Пример

Возможным действиям робота в **ABSTRIPS** соответствуют операторы. Каждый оператор ***Pi*** представлен наименованием со списком параметров ***sp***, условиями применимости оператора ***cond*** и преобразованиями ***action***, которые он совершает, изменяя пространство: ***Pi (sp, cond, action)***.

Пространство поиска (конкретное пространство ***K***), в котором ABSTRIPS ищет решение, состоит из возможных состояний мира, получаемых преобразованием исходного состояния путем применения к нему всех возможных операторов.

3. Метод нисходящего уточнения. Пример

Для того чтобы упростить процесс решения задачи, ABSTRIPS формирует из конкретного пространства *иерархию абстрактных пространств*.

В ABSTRIPS использован простой и изящный подход для формирования абстрактных пространств из конкретного пространства: абстрактные пространства образуются ***путем упрощения условий применимости операторов***, т.е. чем выше уровень абстракции, тем меньше литер (параметров) содержит условие применимости каждого оператора.

Такой подход позволяет при формировании абстрактного пространства не вычеркивать несущественные детали из описания мира и операторов, а просто не учитывать их при решении.

3. Метод нисходящего уточнения. Пример

Уровень детальности указывается с помощью ***веса***, связанного с каждой литерой (параметром) в условии применимости оператора.

Основанием для назначения веса послужили следующие эвристические соображения:
существование предметов и их свойств (т.е. наличие комнат, дверей, ящиков) **является** с точки зрения построения плана более **важным фактом, чем положение предметов, которые могут передвигаться роботом**, и тем более чем **положение робота**;
поэтому в абстрактном пространстве учитываются только такие важные факты.

3. Метод нисходящего уточнения. Пример

После построения приблизительного плана его детали будут уточняться в более конкретных пространствах.

Планирование в ABSTRIPS начинается с верхнего уровня абстракции. При этом ***уровень веса*** устанавливается на ***максимум***, т.е. рассматриваются только те части условий применимости, которые содержат литеры (параметры) с максимальным весом.

Планирование на каждом уровне осуществляется от целей. Предусловия, чей вес ниже текущего уровня, планировщиком игнорируются (они будут учтены на более низких уровнях).

3. Метод нисходящего уточнения. Пример

После того как план полностью построен на текущем уровне, ***значение уровня веса уменьшается*** и начинается планирование на следующем уровне. ***План***, полученный на предыдущем уровне, рассматривается ***как цель*** следующего уровня.

Например, на предыдущем уровне в плане указано, что робот толкает *"объект 1"*. В более детальном пространстве будет установлено, что для выполнения этого действия робот должен находиться в *"комнате 3"*, в которой находится *"объект 1"*, и, например, может быть сформирован план перехода робота из *"комнаты 2"* в *"комнату 3"*.

4. Принцип наименьших свершений

Основной недостаток метода "нисходящего уточнения" состоит в том, что он не имеет обратной связи. Метод предполагает, что одни и те же решения должны приниматься в одинаковых ситуациях при решении любой задачи.

При решении ряда задач детализация решения, полученного на абстрактном уровне, оказывается невозможна, так как при построении абстрактного плана были опущены детали, препятствующие его уточнению, т.е. требуется пересмотр абстрактного плана (решения).

В подобных ситуациях целесообразно применение ***принципа наименьших свершений*** *(least-commitment principle)*. В соответствии с данным принципом решение не строится сразу до конца на верхних уровнях абстракции. Частичное решение детализируется постепенно, по мере появления информации, подтверждающей возможность решения и вынуждающей принять решение.

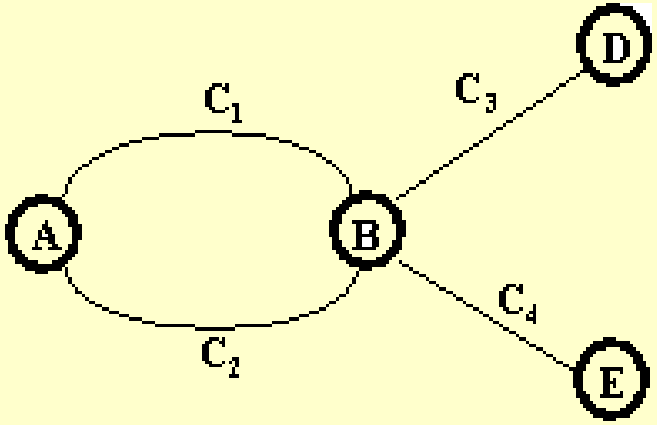
4. Принцип наименьших свершений

Рассуждение, основанное на использовании принципа наименьших свершений, требует, чтобы система была в состоянии:

1) определить, когда накопилось достаточно информации, для принятия решения;
2) приостанавливать работу над некоторой подзадачей, когда для решения нет достаточной информации;
3) переходить с одной подзадачи на другую, возобновляя выполнение приостановленной подзадачи при появлении недостающей информации;
4) объединять информацию, полученную различными подзадачами.

4. Принцип наименьших свершений

Принцип наименьших свершений использует ЭС MOLGEN, предназначенная для планирования экспериментов по молекулярной генетике. MOLGEN представляет взаимодействие между подзадачами в виде ограничений. Для рассуждений об ограничениях используются операторы метауровня (в противовес операторам проблемной области). Система чередует использование принципа наименьших свершений и использование эвристических стратегий.



4. Принцип наименьших свершений

При использовании принципа наименьших свершений MOLGEN делает выбор (принимает решение) только тогда, когда имеющиеся в ее распоряжении ограничения определяют достаточно узкий набор альтернатив. В противном случае процесс решения задачи приостанавливается (задача переходит в состояние *"ожидание больших ограничений"*) и осуществляется переход к другой подзадаче.

4. Принцип наименьших свершений

Распространение ограничений есть механизм для передачи информации между подзадачами. Ограничения, выставленные одной подзадачей, могут существенно сузить набор альтернатив другой подзадачи.
В отличие от ABSTRIPS, осуществляющей построение планов в направлении строго от данных, MOLGEN строит планы в ответ на распространение ограничений.

Чередование в MOLGEN подхода наименьших свершений и эвристических стратегий иллюстрирует ограниченность принципа наименьших свершений. В связи с тем, что любой решатель имеет неполные знания о проблеме, в процессе использования принципа наименьших свершений может возникнуть следующая ситуация: необходимо делать выбор, но нет оснований предпочесть одну альтернативу другим.

4. Принцип наименьших свершений

Эта ситуация приводит к остановке процесса и называется тупиком, потому что все подзадачи перешли в состояние "ожидание больших ограничений". Когда MOLGEN распознает эту ситуацию, она переключается на эвристическую стратегию и делает предположение ("угадывание").

Во многих случаях "угадывание" позволяет продолжить процесс поиска решения и довести его до конечного результата. В других случаях "угадывание" приводит к конфликтам, требующим новых попыток по угадыванию. Конфликт может возникнуть и при работе по принципу наименьших свершений, а именно, в том случае, когда цели принципиально недостижимы.

4. Принцип наименьших свершений

Таким образом, принцип наименьших свершений координирует процесс поиска решения с наличием необходимой информации и в соответствии с доступной информацией перемещает фокус активности по решению задачи от одной подзадачи к другой.

Данный подход непригоден, когда существует много возможностей, но нет надежных оснований для выбора решения. В этих случаях необходимо использовать некоторые формы правдоподобных рассуждений или переходить на использование другой модели.

Необходимо отметить, что для управления процессом решения задачи принцип наименьших свершений требует больших знаний, чем метод нисходящего уточнения.