Интеллектуальные системы

Лекция 3. Представление знаний

Хачумов М.В., к.ф.-м.н.,
доцент кафедры информационных технологий

Представление знаний

Представление знаний — вопрос, возникающий в когнитологии (науке о мышлении), в информатике и в исследованиях искусственного интеллекта. Одна из проблем в представлении знаний — как хранить и обрабатывать знания в информационных системах формальным способом так, чтобы машины могли использовать их для достижения поставленных задач. Примеры применения— экспертные системы, машинный перевод, компьютеризированное техническое обслуживание и системы извлечения и поиска информации (включая пользовательские интерфейсы баз данных).

Данные – отдельные факты, характеризующие объекты, процессы и явления в предметной области, а также их свойства.

Знания – это закономерности предметной области, полученные в результате практической деятельности и опыта, позволяющие специалистам ставить и решать задачи в этой области.

Отличие знаний от данных: Знания являются более сложной категорией информации по сравнению с данными. Знания описывают не только отдельные факты, но и взаимосвязи между ними, поэтому знания иногда называют структурированными данными. Знания могут быть получены на основе обработки эмпирических данных.

Представление знаний

Декларативные знания - знания, которые записаны в памяти интеллектуальной системы так, что они непосредственно доступны для использования после обращения к соответствующему полю памяти. Обычно декларативных знаний используются для представления информация о свойствах и фактах предметной области.

Процедурные знания - знания, хранящиеся в памяти интеллектуальной системы в виде описаний процедур, с помощью которых их можно получить. Обычно процедурные знания используются для представления информации о способах решения задач в проблемной области, а также различные инструкции, методики и т.п.

Лингви́стика (от лат. lingua — язык), языкозна́ние, языкове́дение — наука, изучающая языки. Это наука о естественном человеческом языке вообще и обо всех языках мира как его индивидуализированных представителях.

Сема́нтика (от др.-греч. σημαντικός «обозначающий») — раздел лингвистики, изучающий смысловое значение единиц языка. В качестве инструмента изучения применяют семантический анализ. В конце XIX — начале XX века семантика часто называлась также семасиоло́гией (от др.-греч. σημασία «знак; указание»). Учёные, занимающиеся семантикой, до сих пор обычно называются семасиологами. Также «семантикой» может обозначаться сам круг значений некоторого класса языковых единиц (например, «семантика глаголов движения»).

Семантические сети

Для представления знаний можно использовать **семантические сети**. Каждый узел такой сети представляет концепцию, а дуги используются для определения отношений между концепциями.

Понятие семантической сети возникло в 1966 г. году в работах М.Р. Квиллиана при попытке описания семантики глагола с помощью графа специального вида. Это описание было составлено из вершин, в которых находились лексические единицы анализируемого предложения и «ассоциативных» дуг, служащих для описания ссылок одних вершин на другие. Для таких описаний М.Р. Квиллиан ввел термин «семантическая память». Каждой вершине в семантической памяти соответствовала некоторая «страница», содержащая определение соответствующего вершине понятия. Каждый из указателей относился к одному из следующих *типов: подкласс, дизъюнкция, конъюнкция, свойство, субъект.*

Такие структуры обладали некоторыми дедуктивными свойствами, порождаемыми отношениями «подкласс» и «свойство». Один из механизмов вывода в семантической сети Квиллиана состоял в распространении активности. Пути от начальных вершин к общей вершине определяют некоторое отношение между двумя лексическими единицами. Иначе говоря, речь шла об обнаружении неявно (имплицитно) заданной информации для дальнейшего ее использования в интеллектуальной системе.

Семантические сети

Роберт Ковальский из Эдинбурга в 1979 г. ввел понятия *простых* и *расширенных* семантических сетей, использовав клаузальную логику для их определения.

Для рассмотрения семантических сетей такого вида вернемся к языку исчисления предикатов первого порядка, а именно, к его клаузальной форме.

Клауза есть выражение вида B1 , B2 ,..., Bm ← A1,...,An

где B1 , B2 ,..., Bm суть атомарные формулы, n≥0 и m≥0.

Атомарные формулы A1,...,An суть совместные посылки клаузы, а B1 , B2 ,..., Bm суть альтернативные заключения. (Множество клауз совместно, если оно истинно в одной из моделей языка).

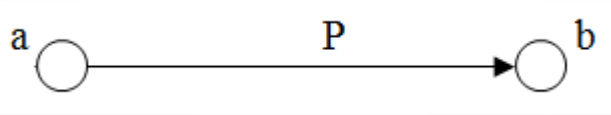
Если клауза содержит переменные x1, x2,…,xk, то она соответствует формуле с квантором всеобщности: ∀x1, ∀x2 ,…, ∀xk (B1 ∨ B2 ∨... ∨ Bm, если A1^,…,^An.)

Если n=0, то клаузу следует интерпретировать так:

∀x1, ∀x2,…, ∀xk, (B1 ∨ B2 ∨... ∨ Bm.). Если m=0, то интерпретация такова: ¬∃ x1, ¬∃x2,…, ¬∃xk A1^,…,^ An.

Если n=m=0 то клауза является тождественно ложным высказыванием и записывается □.

Простая семантическая сеть может рассматриваться как форма записи утверждений клаузальной логики без свободных переменных. Например, клауза P(a,b) ← в языке простых семантических сетей изображается как дуга, помеченная меткой P и направленная из a в b:

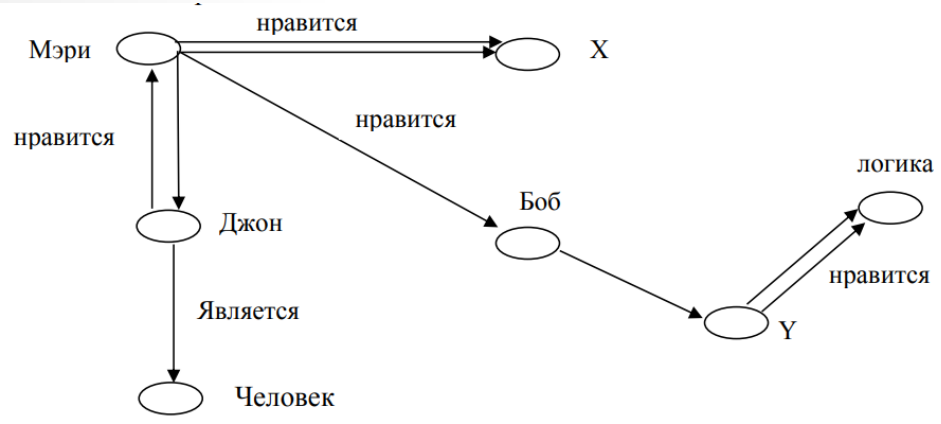


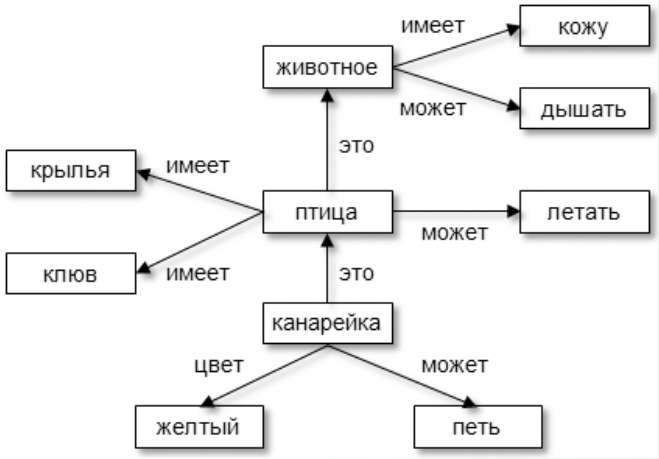
Семантические сети

В расширенных семантических сетях, как и в простых, вершины сопоставляются индивидам, а ребра – бинарным отношениям.

Однако, вершины в расширенных семантических сетях могут соответствовать константным символам, переменным или термам. Атомарные формулы, соответствующие условиям клауз описываются с помощью двойных дуг, а заключения – одинарных

Например, расширенная семантическая сеть на рис





соответствует множеству клауз:

Джону нравится Мери ←

Джон является человеком←

Мери нравится Джон, Мери нравится Боб←Мери нравится x

Бобу нравится y←y нравится логика

Они позволяют интегрировать в одном представлении синтаксис и семантику (т.е. интерпретацию) клаузальных форм. Это позволяет в процессе вывода обеспечивать взаимодействие синтаксических и семантических, теоретико-модельных подходов.

Продукционная модель знания

Продукционные модели - это наиболее распространенные на текущий день модели представления знаний, где знания описываются с помощью правил «если-то» (явление → реакция) и представляются а виде:

ЕСЛИ условие (антецедент)

ТО действие (консеквент)

Под условием понимается некоторое предложение-образец, по которому осуществляется поиск в базе знаний, а под действием – набор действий, выполняемых при успешном исходе поиска.

При использовании таких моделей у систем, основанных на знаниях, имеется возможность:

- применение простого и точного механизма использования знаний;
- представления знаний с высокой однородностью, описываемых по единому синтаксису.

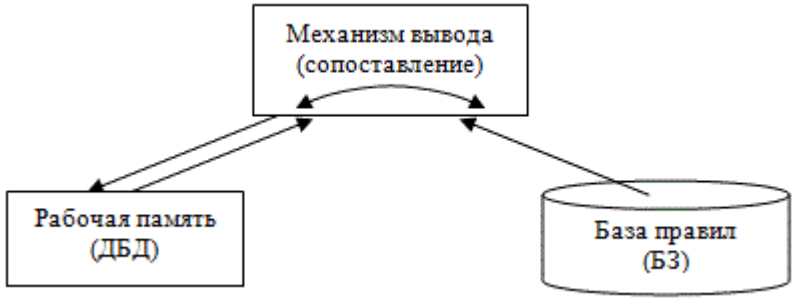
Программные средства, оперирующие со знаниями, представленными правилами, получили название продукционных систем и впервые были предложены Постом в 1941 году.

На самом высоком уровне их описания можно выделить следующие компоненты: ***рабочую память*** (или, как иногда говорят, глобальную базу данных), ***множество правил***, выполняющих некоторые действия (во внешней среде и в рабочей памяти) и некоторую ***стратегию управления***, в соответствии с которой происходит выбор правил для применения и выполнение действий.

Продукционная модель знания

Общим для систем продукций является то, что они состоят из трех основных элементов:

- Набора правил, используемых как база знаний (БЗ), который чаще всего называют базой правил.
- Рабочей памяти, где хранятся предпосылки, касающиеся отдельных задач, а также результаты выводов, получаемых на основе этих предпосылок (динамическая база данных - ДБД).
- Механизма логического вывода, использующего правила в соответствии с содержимым рабочей памяти.



Правила применяются к рабочей памяти. В состав каждого правила входит некоторое ***условие***, которому текущее состояние рабочей памяти может удовлетворять, либо нет. Правило может быть применено, если условие выполнено. Применение правила изменяет состояние рабочей памяти. Стратегия управления выбирает, какое именно правило из числа применимых следует использовать и прекращает вычисления, когда состояние рабочей памяти удовлетворяет целевому условию.

Продукционная модель знания

С точки зрения архитектуры такой подход обладает следующими существенными отличиями от архитектур традиционных программных систем:

- рабочая память доступна всем правилам;
- отсутствуют вызовы правил из других правил;
- отсутствует априорно заданный алгоритм решения задачи (т.е. порядок выполнения правил) – алгоритм решения задачи является одним из результатов её решения;
- данные и результаты вычислений становятся доступными правилам только через рабочую память.

Особенности организации систем, основанных на правилах, как легко видеть, обеспечивают, в значительной степени, модульный их характер и изменения в рабочей памяти, множестве правил или в стратегии управления могут проводиться относительно независимо. Эти свойства систем, основанных на правилах, хорошо согласуются с эволюционным характером разработки больших программных систем, предполагающих использование значительных объемов знаний.

Продукционная модель знания

Определение. Правилом называется упорядоченная тройка множеств П = <C, A, D> , где

- C – условие правила;
- A – множество добавляемых правилом фактов;
- D -множество удаляемых правилом фактов.

Для записи элементов основных конструкций языка представления знаний (в данном случае, языка правил), т.е. условия C правила П, множеств A и D будем использовать язык исчисления предикатов первого порядка. А именно, будем полагать, что каждое из упомянутых множеств есть множество атомарных формул языка исчисления предикатов первого порядка.

Фактами называются атомарные формулы исчисления предикатов первого порядка без свободных переменных.

Будем считать, что в правилах атомарные формулы из множеств C, A и D превращаются в факты в процессе применения правила, т.е. в результате выполнения соответствующих подстановок (m_1 , m_2 ,…, m_n) на места свободных переменных (x_1 ,x_2 ,…,x_n) и проверки для каждой формулы $P(x_1, x_2,…,x_n)$ из C условия $(m_1, m_2,…, m_n) \in I(P)$, т.е. выполнимости в текущем состоянии рабочей памяти.

Механизм функционирования систем продукции

Допустим, что данные, записанные в рабочую область, представляют собой образцы в виде набора символов:

«намерение – отдых»

«место отдыха – горы»

Эти образцы соответствуют фактам «намерение IS отдых» и «место отдыха IS горы», которые при их записи в предикатной форме имели бы вид: IS(«намерение», «отдых») и IS(«место отдыха», «горы»).

Что касается правил в системах продукций, то они отражают содержимое рабочей памяти. В условной части любого правила находятся:

- либо одиночные образцы,
- либо несколько условий, соединенных предлогом «И».

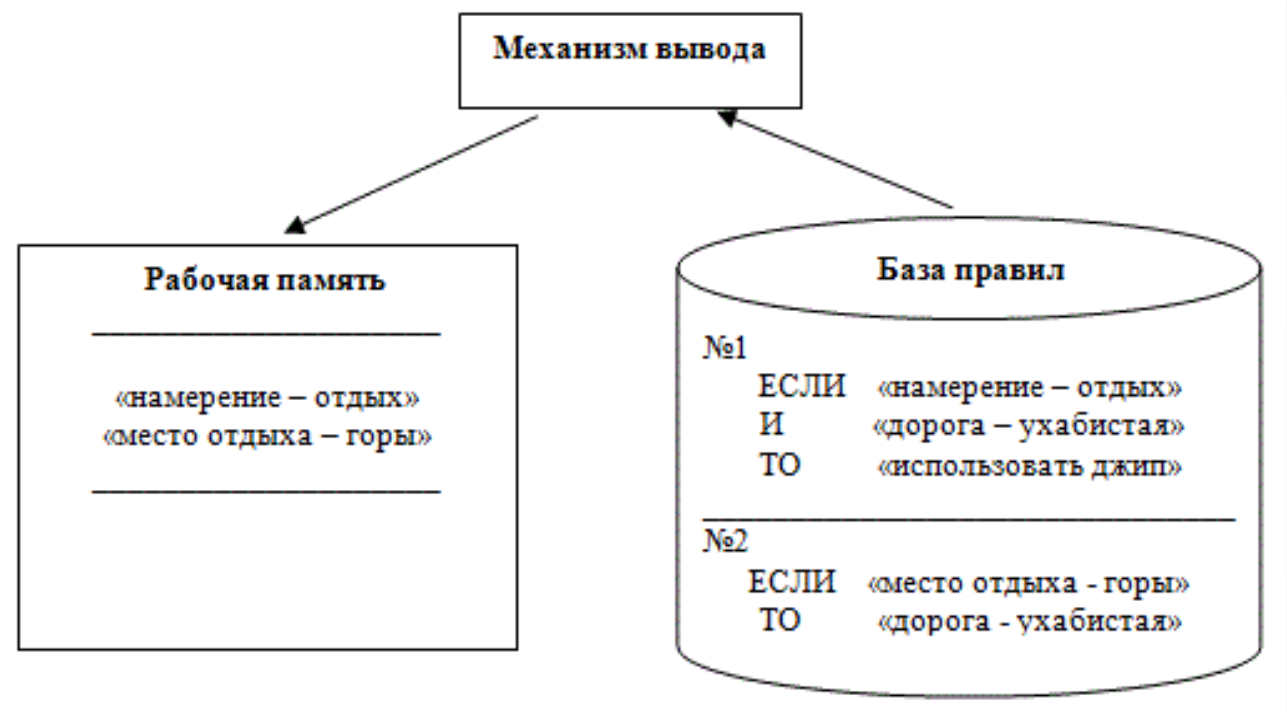
Заключительную часть правил представляют собой образцы, которые система будет дополнительно регистрировать в своей рабочей памяти при возможности сопоставление условной части правила с текущим содержимым рабочей памяти системы.

Предположим, что для описания рассматриваемой предметной области используются всего два правила:

ЕСЛИ «намерение – отдых» <-- правило №1
И «дорога - ухабистая»
ТО «использовать - джип»

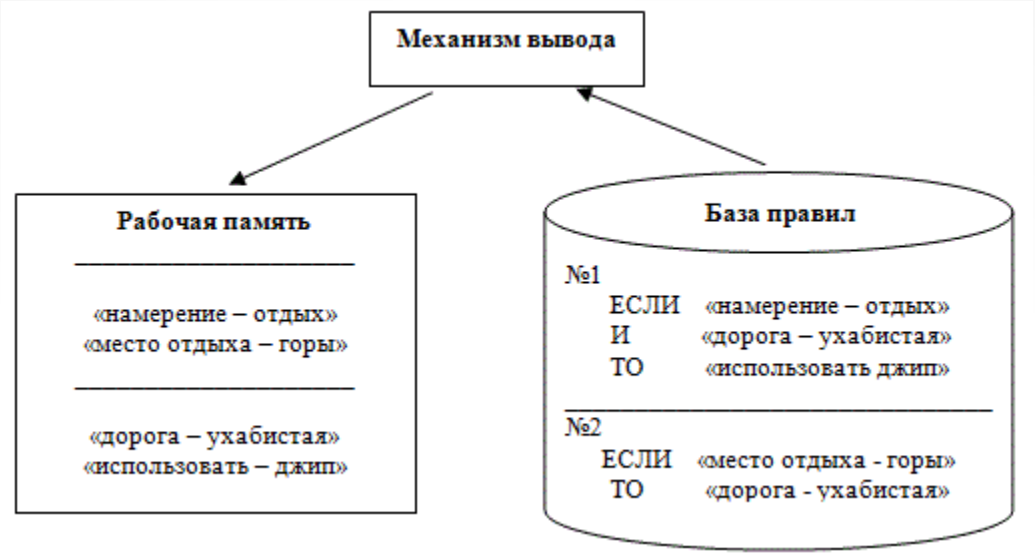
ЕСЛИ «место отдыха – горы» <-- правило №2
ТО «дорога – ухабистая»

Механизм функционирования систем продукции



После того, как в рабочую память будут записаны все исходные образцы (факты), а в базу правил – все правила, описывающие исследуемую предметную область, у системы продукций появляется возможность в использовании встроенного в нее механизма вывода, который будет рассматривать возможность применения этих правил.

Прямая цепочка рассуждений



Если обратиться к рабочей памяти, то исходя из посылок что
«намерения – отдых»
«место отдыха – горы»
результатом вывода является рекомендация
«использовать – джип»
с пояснением причин данного вывода, которая определяется тем, что
«дорога – ухабистая»
В процессе реализации алгоритма механизма логического вывода для данного примера проводилась работа по:

- многократному просмотру содержимого базы правил;
- последовательному применению правил на основе предварительно записанного содержимого рабочей памяти;
- дополнению данных, помещаемых в рабочую память.

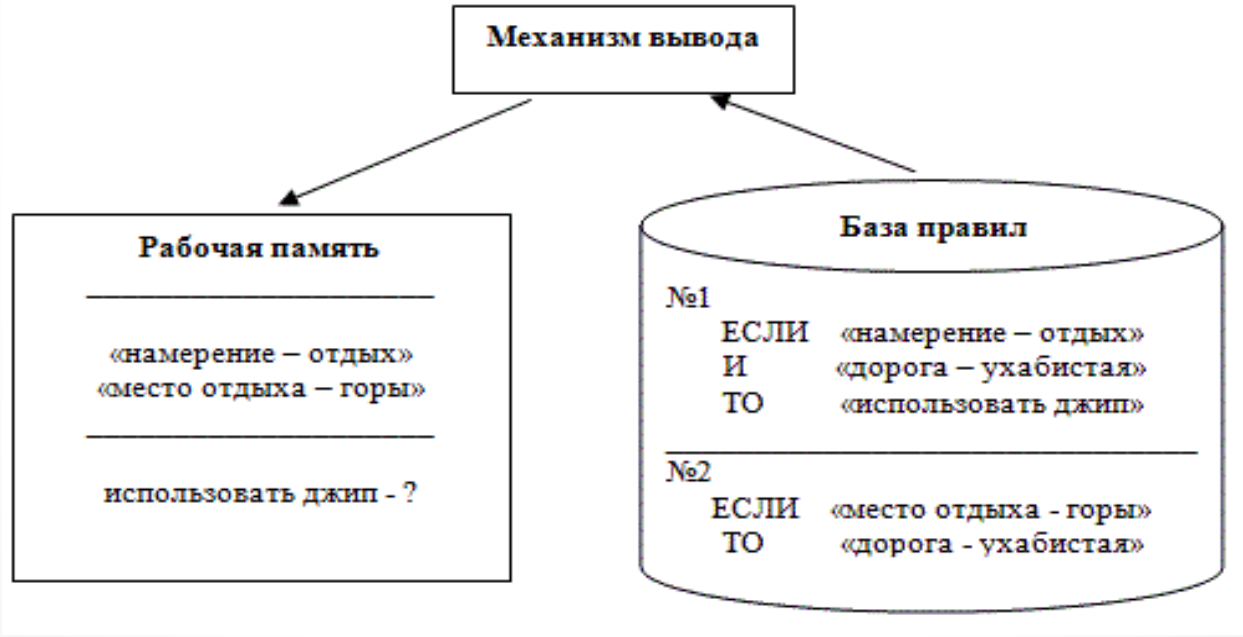
Прямая цепочка рассуждений

После запуска в работу рассматриваемого примера системы продукции алгоритм ее работы и последовательность логического вывода будет следующей:

1. Механизм вывода анализирует правила, начиная с первого, определяет наличие образца «намерение – отдых» в рабочей памяти и отсутствие в ней образца «дорога – ухабистая».
2. Условная часть правила №1 считается ложной, и механизм вывода переходит к следующему правилу (в нашем случае к правилу №2).
3. Условная часть правила №2 признается истинной, т.к. образец «место отдыха – горы» присутствует в рабочей памяти и механизм вывода переходит к выполнению его заключительной части.
4. Заключительная часть правила №2 «дорога – ухабистая» заносится в рабочую память.
5. После просмотра всех правил происходит вторичное их применение, начиная с первого правила, за исключением тех, которые уже были применены (в примере это правило №2).
6. При повторном сопоставлении правила №1 его условная часть становится истинной ввиду доопределения рабочей памяти, и механизм вывода выполняет его заключительную часть.
7. Заключительная часть «использовать - джип» переносится в рабочую память, а правило №1 исключается из дальнейшего согласования.
8. Правил для сопоставления не остается, и система останавливается.

Обратная цепочка рассуждений

Считая, что рабочая память содержит образцы
«намерения – отдых» «место отдыха – горы»,
а база правил содержит оба, отмеченных выше правила, то целью составления (логического вывода) является доказательство факта
«использовать – джип».
Таким образом, рабочая память и база правил будет иметь исходный вид, аналогичный тому, что приведен на рис.



Результатом вывода является подтверждение цели «использовать – джип» при условии, что «дорога – ухабистая».

Обратная цепочка рассуждений

Последовательность составления системой продукции следующая:

1. Определяется правило, в котором в заключительной части содержится целевой факт.
2. Исследуется возможность применения первого правила для подтверждения исходного факта.
3. Поскольку образец «намерение – отдых» из условной части правила №1 занесен в рабочую память, то для достижения цели достаточно подтвердить факт «дорога – ухабистая».
4. Образец «дорога – ухабистая» принимается за новую цель, и необходимо найти правило, подтверждающее этот факт.
5. Исследуется возможность применения правила №2. Условная часть этого правила является истинной, т.к. образец «место отдыха – горы» имеется в рабочей памяти;
6. В виду возможности применения правила №2, рабочая память пополнится образцом «дорога – ухабистая» и появляется возможность применения правила №1 для подтверждения цели «использовать – джип».

Механизм функционирования систем продукции

На глобальном уровне управления последовательностью применения правил обычно выделяют две основные стратегии — применять правила в прямом или обратном порядке.

1. Прямой порядок означает, что цепь рассуждений строится, отталкиваясь от данных (условий, о которых известно, что они удовлетворяются), к гипотезам (состоянию проблемы, вытекающему из этих условий).

2. Обратная цепочка означает, что рассуждения строятся, отталкиваясь от заданной цели (гипотезы, представляющие целевое состояние системы) к условиям, при которых возможно достижение этой цели.

На практике наиболее часто встречаются механизмы логического вывода, опирающиеся на обратную цепочку рассуждений. Это обусловлено их более надежной работой (практически всегда имеется возможность найти цепочку рассуждений от конца до начала) и большей производительностью, что становится особенно заметно при большом количестве продукций.

<условие> <действие> <дополняемы факты> <удаляемые факты>

Так работают 1) системы интеллектуального управления 2) системы интерпретации данных 3) системы диагностики 4) системы планирования 5) системы прогнозирования 6) системы мониторинга

Интеллектуальные системы

Практическое занятие №2

Хачумов М.В., к.ф.-м.н.,
доцент кафедры информационных технологий

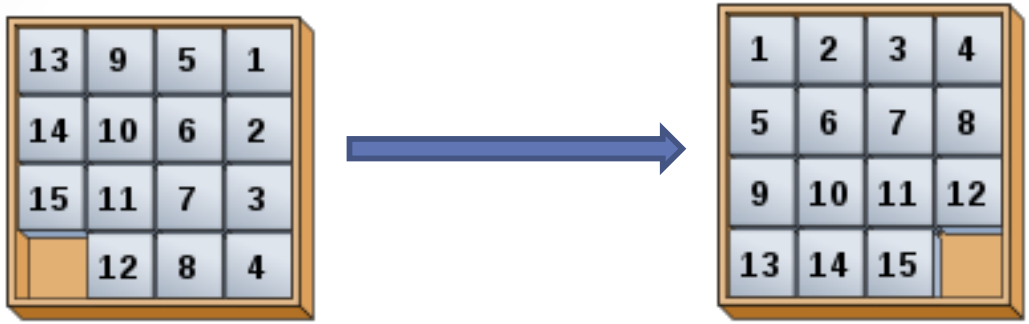
Игра в 15

Игра в 15, пятнашки, такен — популярная головоломка, придуманная в 1878 году Ноем Чепмэном. Представляет собой набор одинаковых квадратных костяшек с нанесёнными числами, заключённых в квадратную коробку. Длина стороны коробки в четыре раза больше длины стороны костяшек для набора из 15 элементов, соответственно в коробке остаётся незаполненным одно квадратное поле. Цель игры — перемещая костяшки по коробке, добиться упорядочивания их по номерам, желательно сделав как можно меньше перемещений.

Математика перестановок

В головоломке The Gem Puzzle, которую в 1879 году производил и продавал Матиас Райс, игрок высыпал плитки из коробочки и случайным образом укладывал их обратно, после чего пытался восстановить упорядоченную конфигурацию:

Поместите блоки в коробочку беспорядочным образом, затем перемещайте, пока они не выстроятся в правильном порядке.



В другой версии все плитки, за исключением 14 и 15, изначально находятся на своих местах; задача состоит в том, чтобы поменять местами неправильно стоящие плитки 14 и 15. Эта задача неразрешима; тем не менее, в этом случае можно упорядочить плитки с пустой ячейкой в верхнем левом углу либо выстроить фишки столбцами

Начальное положение

Недостижимое расположение

Достижимое расположение

Математика перестановок

Количество возможных начальных положений пятнашек равно n!

Размер	Число начальных положений
2 × 2	24
2 × 4	40 320
3 × 3	362 880

Можно показать, что ***ровно половину*** из всех возможных начальных положений пятнашек ***невозможно*** привести к собранному виду.

Пусть квадратик с числом ***i*** расположен до (если считать слева направо и сверху вниз) ***k*** квадратиков с числами меньшими ***i***. Будем считать $n_i=k$, то есть если после костяшки с i-м числом нет чисел, меньших i, то $k=0$. Также введем число e — номер ряда пустой клетки (считая с 1). Если сумма

$$N = \sum_{i=1}^{15} n_i + e$$

является нечётной, то решения головоломки не существует

N=1+4=5

N=0+4=4

N=0+1=1

Математика перестановок

Размер	Число решаемых конфигураций	«Число Бога»
2 × 2	12	6
2 × 3	360	21
2 × 4	20 160	36
2 × 5	1 814 400	55
2 × 6	239 500 800	80
2 × 7	43 589 145 600	108
2 × 8	10 461 394 944 000	140
3 × 3	181 440	31
3 × 4	239 500 800	53
3 × 5	653 837 184 000	84
4 × 4	10 461 394 944 000	80
5 × 5	7,7556·10^24	208

Для обобщённых пятнашек (с бо́льшим, чем 15, количеством костяшек) задача поиска кратчайшего решения для заданной конфигурации является ***NP-полной***

Эвристика

Эвристический алгоритм (эвристика) — алгоритм решения задачи, включающий практический метод, не являющийся гарантированно точным или оптимальным, но достаточный для решения поставленной задачи. Позволяет ускорить решение задачи в тех случаях, когда точное решение не может быть найдено.

Эвристический алгоритм — это алгоритм решения задачи, правильность которого для всех возможных случаев не доказана, но про который известно, что он даёт достаточно хорошее решение в большинстве случаев. В действительности может быть даже известно (то есть доказано) то, что эвристический алгоритм формально неверен. Его всё равно можно применять, если при этом он даёт неверный результат только в отдельных, достаточно редких и хорошо выделяемых случаях или же даёт неточный, но всё же приемлемый результат.

Проще говоря, ***эвристика*** — это не полностью математически обоснованный (или даже «не совсем корректный»), но при этом практически полезный алгоритм.

Важно понимать, что эвристика, в отличие от корректного алгоритма решения задачи, обладает следующими особенностями.

- Она не гарантирует нахождение лучшего решения.
- Она не гарантирует нахождение решения, даже если оно заведомо существует (возможен «пропуск цели»).
- Она может дать неверное решение в некоторых случаях.

Эвристика

Одна из самых простых эвристик для пятнашек: *число ходов, требуемых для решения, не меньше, чем число плиток, находящихся не на своих местах.*

Верность утверждения, то есть допустимость эвристической функции «число плиток, находящихся не на своих местах», следует из того, что за один ход может быть поставлена на место только одна плитка.

Более «умная» эвристика сопоставляет каждому расположению плиток сумму расстояний от текущей позиции каждой плитки до её целевой позиции. В литературе эта эвристика встречается под именем «манхэттенское расстояние» (Manhattan distance). Допустимость функции следует из того, что за один ход перемещается только одна фишка, и расстояние между этой фишкой и её конечной позицией изменяется на 1. Тем не менее, эта эвристика также не использует всю имеющуюся информацию, из-за того, что в одной позиции не могут находиться одновременно две плитки.

H=0
M=0

H=2
M=2

H=15
M=24

Задание (поиск решения в глубину)

Входные данные

int M_g[n][n] = { {1, 2, 3}, {8, 0, 4}, {7, 6, 5} }; //целевая матрица

int M_i[n][n] = { {5, 3, 4}, {6, 0, 7}, {8, 2, 1} }; // начальная матрица

```
left_rule() {   //правила перестановок
            int tmp = M1[x-1][y];
            M1[x-1][y]=M1[x][y];
            M1[x][y] = tmp;
}
right_rule(); up_rule(); down_rule().
```

Выбор очередного правила может осуществляться: либо случайным образом либо с применением эвристики

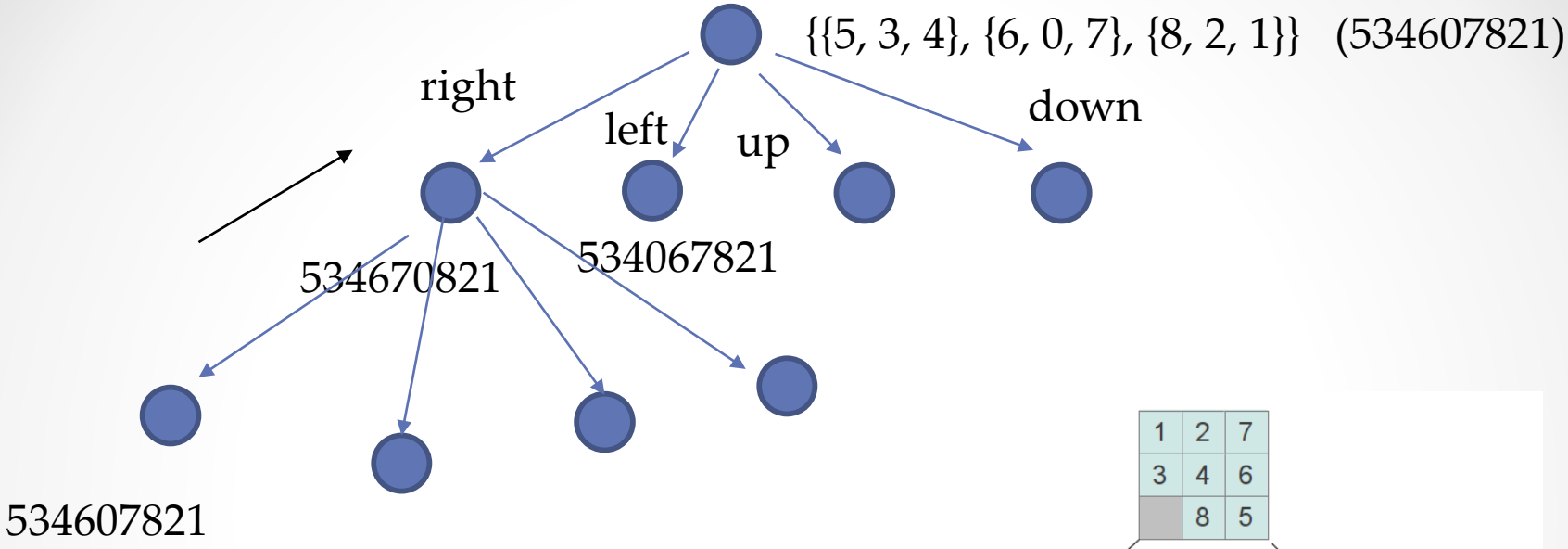
Выходные данные

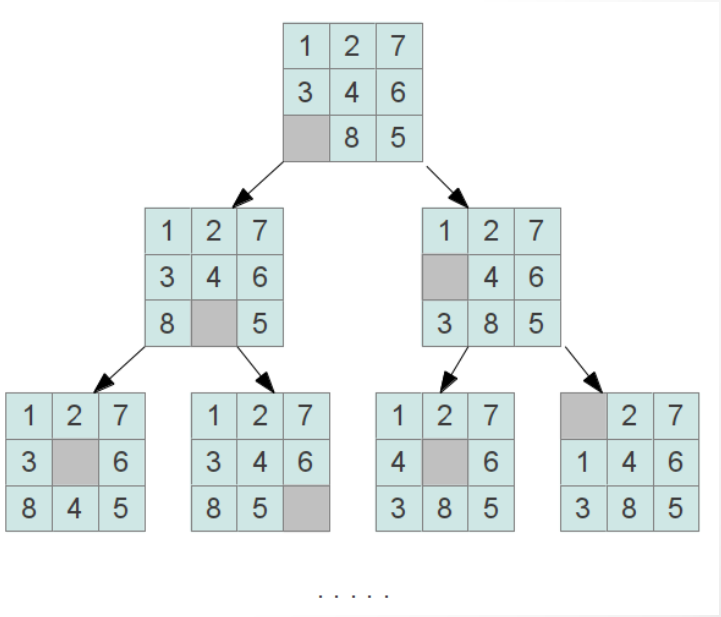
Распечатываем все шаги, расстояние до целевой матрицы, саму матрицу, глубину поиска

```
Step: 710 Deep: 622 Distance: 5 Matrix: 482103765
Step: 711 Deep: 623 Distance: 6 Matrix: 482013765
Step: 712 Deep: 624 Distance: 6 Matrix: 082413765
Step: 713 Deep: 625 Distance: 6 Matrix: 802413765
Step: 714 Deep: 626 Distance: 5 Matrix: 812403765
Step: 715 Deep: 627 Distance: 6 Matrix: 812043765
Step: 716 Deep: 628 Distance: 5 Matrix: 012843765
Step: 717 Deep: 629 Distance: 4 Matrix: 102843765
Step: 718 Deep: 630 Distance: 3 Matrix: 142803765
Step: 719 Deep: 631 Distance: 4 Matrix: 142830765
Step: 720 Deep: 632 Distance: 4 Matrix: 140832765
Step: 721 Deep: 633 Distance: 4 Matrix: 104832765
Step: 722 Deep: 634 Distance: 3 Matrix: 134802765
Step: 723 Deep: 635 Distance: 4 Matrix: 134820765
Step: 724 Deep: 636 Distance: 3 Matrix: 130824765
Step: 725 Deep: 637 Distance: 2 Matrix: 103824765
Step: 726 Deep: 638 Distance: 0 Matrix: 123804765
Marix is found! n=0my matrix
1 2 3
8 0 4
7 6 5

--------------------------------
Process exited after 0.9505 seconds with return value 0
Для продолжения нажмите любую клавишу . . .
```

Задание (поиск решения в глубину)





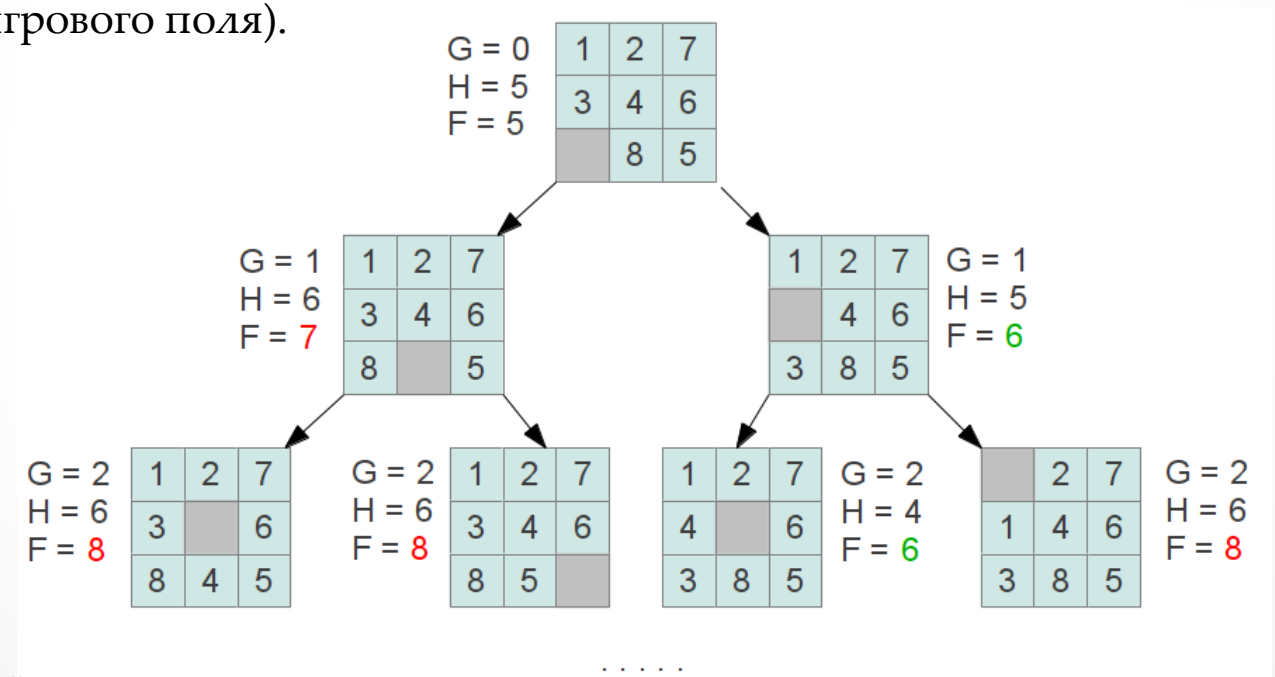
```
void pyatnashki (int deep) {
//проверка условий
…..
//эвристика
//применение правила
pyatnashki(deep);
}
```

На максимальное количество баллов надо: **придумать свою эвристику, учитывать состояния в которых мы уже были (использовать рекурсию)**

Задание (алгоритм A*)

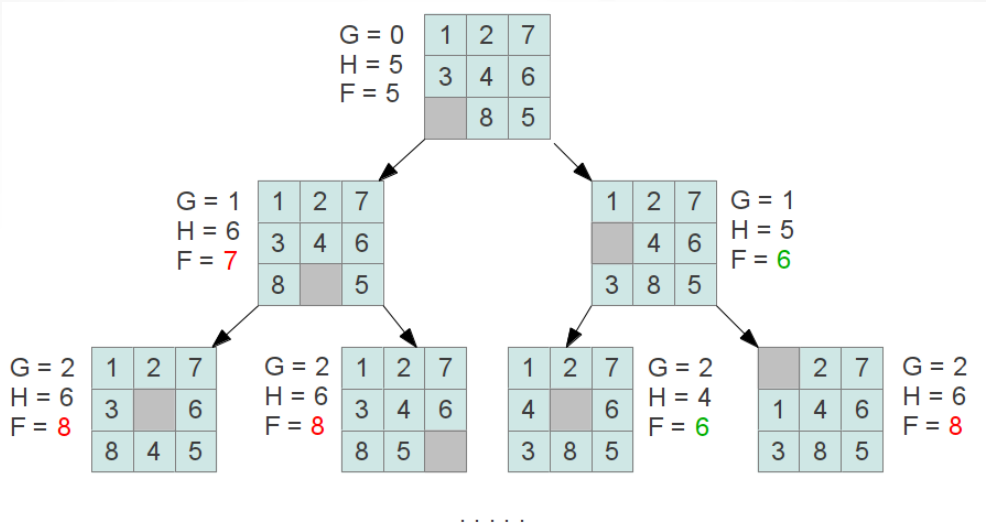
Алгоритм A* предполагает наличие двух списков вершин графа: открытого и закрытого. В первом находятся вершины, еще не проверенные алгоритмом, а во втором те вершины, которые уже встречались в ходе поиска решения.

На каждом новом шаге, из списка открытых вершин выбирается вершина с наименьшим весом. Вес (F) каждой вершины вычисляется как сумма расстояния от начальной вершины до текущей (G) и эвристическое предположение о расстоянии от текущей вершины, до терминальной (H). Fi = Gi + Hi, где i - текущая вершина (состояние игрового поля).



Для "Пятнашек" можно сделать предположение, что для достижения терминальной вершины, необходимо выполнить перемещений не меньше, чем количество костяшек, находящихся не на своих местах, а расстояние от начальной вершины до текущей рассчитывать как количество сделанных перестановок

Задание (алгоритм A*)



Для выбранной вершины порождаются дочерние вершины (состояния, которые могут быть получены перемещением костяшек на пустую клетку). Каждая вершина имеет ссылку на родительскую, т.е. "помнит" из какого состояния в нее перешли.

Выполняется перебор дочерних вершин. Каждая дочерняя вершина проверяется на предмет наличия в списке закрытых. Если вершина не встречалась ранее, она перемещается в список открытых вершин. Для нее рассчитывается эвристическое расстояние до терминальной вершины и пересчитывается расстояние от начальной вершины, ведь есть вероятность, что текущий путь к этой вершине окажется короче, чем найденный ранее (текущая вершина могла уже находится в списке открытых). Если путь оказался короче, ссылка на родительскую вершину изменяется.

Последовательность действий повторяется, пока в списке открытых вершин есть хотя бы одна вершина или пока в ходе выполнения алгоритма не встретится терминальная вершина.

Задание (алгоритм A*)

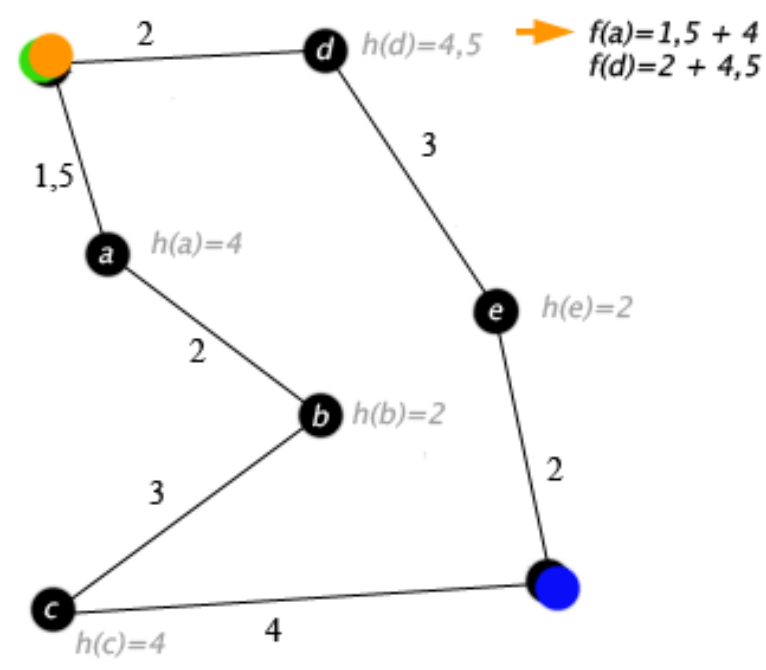
- Q — множество вершин, которые требуется рассмотреть,
- U — множество рассмотренных вершин,
- f[x] — значение эвристической функции "расстояние + стоимость" для вершины x,
- g[x] — стоимость пути от начальной вершины до x,
- h(x) — эвристическая оценка расстояния от вершины x до конечной вершины.

На каждом этапе работы алгоритма из множества Q выбирается вершина с наименьшим значением эвристической функции и просматриваются её соседи. Для каждого из соседей обновляется расстояние, значение эвристической функции и он добавляется в множество Q.

Псевдокод:

```
bool A*(start, goal):
    U = ∅
    Q = ∅
    Q.push(start)
    g[start] = 0
    f[start] = g[start] + h(start)
    while Q.size() != 0
        current = вершина из Q с минимальным значением f
        if current == goal
            return true // нашли путь до нужной вершины
        Q.remove(current)
        U.push(current)
        for v : смежные с current вершины
            tentativeScore = g[current] + d(current, v) // d(current, v) — стоимость пути между current и v
            if v∈U and tentativeScore >= g[v] continue
            if v∉U or tentativeScore < g[v]
                parent[v] = current
                g[v] = tentativeScore
                f[v] = g[v] + h(v)
                if v∉Q
                    Q.push(v)
return false
```

Пример алгоритма A* + Задание



Входные данные

int M_g[n][n] = { {1, 2, 3}, {8, 0, 4}, {7, 6, 5} }; //целевая матрица

int M_i[n][n] = { {5, 3, 4}, {6, 0, 7}, {8, 2, 1} }; // начальная матрица

Выбор очередного шага осуществляться с применением эвристики

Выходные данные

Распечатываем все шаги, кратчайший путь до целевой матрицы, считаем его длину

1 шаг:
Q = a, d; U=0.
f(a) = 1,5 + 4 = **5,5**
f(d) = 2 + 4,5 = 6,5

2 шаг:
Q = d, b; U=a.
f(b) = 1,5 + 2 + 2= **5,5**
f(d) = 2 + 4,5 = 6,5

3 шаг:
Q = d, c; U=a, b.
f(c) = 1,5 + 2 + 3 + 4= 10,5
f(d) = 2 + 4,5 = **6,5**

4 шаг:
Q = e, c; U=a, b, d.
f(c) = 1,5 + 2 + 3 + 4= 10,5
f(e) = 5 + 2 = **7**

5 шаг:
Встретилась терминальная вершина

Start->d->e->End
G = 2 + 3 + 2 = 7