Grundzüge der Artificial Intelligence
TU WIEN 2008WS
G.B.KAYIK
0627591

Index

QUESTIONS

1.SEARCH

1:1 Ein Suchbaum war gegeben. Man sollte mit den 5 bekannten uninformierten Suchalgos. Die Pfadkosten und den Zielknoten angeben.

1:2Wann ist eine Heuristik admissible / Wann dominiert eine Heuristik h1 eine Heuristik h2

*A heuristic function h(n) is called **admissible** if for all nodes one has h(n) < k(n) where k(n) is the actual distance to the goal from n. Both of the heuristics given above are admissible.*

If all the costs are positive and the heuristic is admissible then A terminates and finds the shortest path. Like breadth first search A* can use a lot of memory. If we combine the iterative deeepening idea with A* one gets an algorithm called IDA* (for iterative deepening A*). The space is searched depth first for successively larger bounds on the heurisitic function f(n).*

In computer science, a heuristic is said to be **admissible** if it is no more than the lowest-cost path to the goal. In other words, a heuristic is admissible if it never overestimates the cost of reaching the goal.

n is a node
h is a heuristic
$h(n)$ is cost indicated by h to reach a goal from n
$h^*(n)$ is the actual cost to reach a goal from n

A heuristic function h is **optimistic** or **admissible** if
$h(n) \leq h^*(n)$ for all nodes n.

$$\forall n, h(n) \leq h^*(n)$$

H is called an admissible heuristic . Admissible heuristic by nature optimistic, because they think the cost of solving problem is less than it actually is.
If h2(n) ≥ h1(n) for all n (both admissible)
then h2 dominates h1 and is better for search
when there is given any admissible heuristics ha, hb,
h(n) = max(ha(n), hb(n))
is also admissible and dominates ha, hb

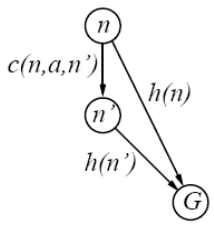
1.3 Beschreibe den Hill-Climbing Algo (Pseudcode)

The Hill Climbing Algorithm is simply a loop that continually moves in the direction of increasing value.

```
function Hill-Climbing( problem) returns a state that is a local maximum
inputs: problem, a problem
local variables: current, a node
neighbor, a node
current←Make-Node(Initial-State[problem])
loop do
neighbor←a highest-valued successor of current
if Value[neighbor] ≤ Value[current] then return State[current]
//Return current node since no better neighbors exist
current←neighbor
end.
```

1.4 Was ist die Consistency bei einer Heuristic.

A heuristic is consistent if
h(n) ≤ c(n, a, n′) + h(n′)
If h is consistent, we have
f(n′) = g(n′) + h(n′)
= g(n) + c(n, a, n′) + h(n′)
≥ g(n) + h(n)
= f(n)
I.e., f(n) is nondecreasing along any path.



1.5 Aus welchem komponenten besteht ein Suchproblem .
A problem consist of four parts as initial state, a set of operators, goal test function and path cost function.

Initial state: that the agent knows itself to be in

Operators: the set of possible action available to the agent. The term of operator is used to denote the description of an action in terms of which state will be reached by carrying out the action in a particular state.

Goal Test: The goal test, which the agent can apply to a single state description to determine to another , even though they both reach the goal. For example in chess, the goal is to reach a state called check mate where the opponents king ca n be captured on the next moved no matter what the opponent does.
Path Cost Function : A path Cost component is a function, that assigns a cost to a path. generally the path function is denoted *g*. the number of action to reach goal

1.7 Was ist die Lösung eines Suchproblems ?
The output of a search algorithm is a solution that is , a path from the initial state to a state that satisfies the goal test.

1.8 Properties of search Strategies:
-Completeness
-Time Complexity
-Space Complexity
-Optimality

1.8 uniformed search strategies
Breadth first search
Uniform – Cost Search
Depth- first Search
Depth – limited search
Iterative deeping Search

Optimalitat für 5 type uninformed such algorithm

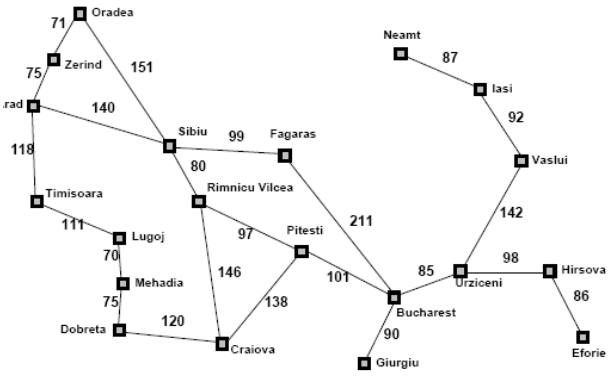
Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Complete?	Yes*	Yes*	No	Yes, if $l \geq d$	Yes
Time	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	b^m	b^l	b^d
Space	b^{d+1}	$b^{\lceil C^*/\epsilon \rceil}$	bm	bl	bd
Optimal?	Yes*	Yes	No	No	Yes*

1.9 unterschied zwischen informed und uninformed Suchstrategie.
Uninformed Search Strategies: the term means that they have no information about the number of steps or the path cost from the current state to the goal. Uninformed search is also called blind search and is less effective than informed search.

Informed search strategies : (**heuristic, intelligent**) use information about the problem (estimated distance from a state to the goal) to guide the search.

INFORMED SEARCH ALGORITHM

1.10 Ein Beispiel mit greedy und A* lösen und angeben ob bei A* die Lösung optimal ist.



GREEDY SEARCH
Let f(n)=h(n) = estimate cost from node n to nearest goal node.
hSLD(n) = straight-line distance from n to Bucharest.
<u>İts neither optimal nor complete</u>
Complete?? No–can get stuck in loops, e.g.,
Iasi → Neamt → Iasi → Neamt →
Complete in finite space with repeated-state checking
Time?? O(bm), but a good heuristic can give dramatic improvement

Space?? O(bm)—keeps all nodes in memory
Optimal?? No

Straight−line distance to Bucharest	
Arad	366
Bucharest	0
Craiova	160
Dobreta	242
Eforie	161
Fagaras	178
Giurgiu	77
Hirsova	151
Iasi	226
Lugoj	244
Mehadia	241
Neamt	234
Oradea	380
Pitesti	98
Rimnicu Vilcea	193
Sibiu	253
Timisoara	329
Urziceni	80
Vaslui	199
Zerind	374

A* SEARCH
Praktisches Beispiel zur A* Suche. ?
The idea behind of the A* avoid expanding paths that are already expensive.

Evaluation Function: f(n)= g(n) + h(n).

g(n) : cost so far to reach n

h(n): estimated cost to goal from n state.

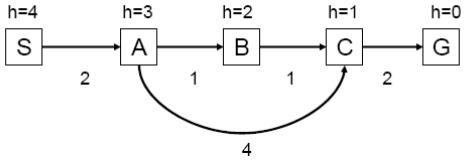
f(n): estimated total cost through n to goal.

- A* search uses an heuristic function.

i.e., h(n) ≤ h*(n) where h*(n) is the true cost from n

- A* search is optimal
- A* search expands lowest g + h
- complete and optimal
- also optimally efficient (up to tie-breaks, for forward search)

praktische Beispel

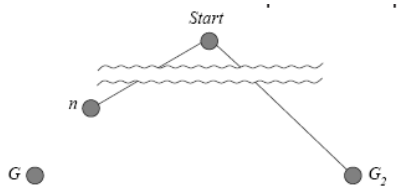


1. Expand S
2. Expand A
3. Choose between B (f(B)=3+2=5) and C (f(C)=6+1=7)) expand B
4. Expand C
5. Expand G - recognize it is the goal

1.11 ist greede und A* complete, optimal?
Greedy Search neither optimal nor complete.

A* is complete and optimal . its optimal because it cannot expand fi+1 until fi is finished

1.12 optimality of A*



$$f(G_2) = g(G_2) \quad \text{since } h(G_2) = 0$$
$$> g(G_1) \quad \text{since } G_2 \text{ is suboptimal}$$
$$\geq f(n) \quad \text{since } h \text{ is admissible}$$

Since $f(G_2) > f(n)$, A* will never select G_2 for expansion

Suppose that supoptimalgoal G2 has been generated and its in the queue and n is unexpanded node on a shortest path to an optimal goal G1.

1.13 Anwenden Greedy algoritmus und Dokumentation von der Eval Funktion.

2.AGENTEN
2. 1 What is an Agent :
An agent is anything that can be viewed as perceiving its environment through sensors and acting through effectors.

Agent include humans agent, robotic agent and software agent.

Four Agententypen

Four basic types of agent programs in order of increasing generality:
– simple reflex agents
– reflex agents with state
– goal-based agents
– utility-based agents

2.2 Aus welchen Bestandteilen besteht ein Agent?

We split an agent in to an architecture on an agent program.
Agent = Architecture + Program
Architecture: programming device + sensors + actuators
Program: gets sensor data, returns actions for the actuators

2.3 Mit welchem Eigenschaften könnte man die Umwelt Beschreiben ?

Four Eigenschaften einer Umgebung angeben/beschreiben

- Completely observable vs. partly observable
  Sensors detect all relevant properties of the world for the current action
- Deterministic vs. stochastic
  Next state determined by the current state and the performed action
- Episodic vs. non-episodic
  Agent's experience is divided into "atomic" parts (indep. from each other)
- Static vs. dynamic
  The world does not change during the reasoning time of the agent
- Discrete vs. continuous

World properties have discrete values (eg time, no of possible states, . . .)

- Single-agent vs. multi-agent
  Only one agent, no cooperation and no competition between agents

2.4 Einen "Agent mit Zustand" aufzeichenen

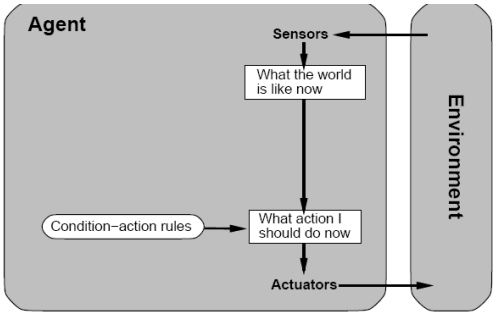


Figure. 2.3 : Simple reflex agent

2.5 Wie wählt ein rationaler Agent seine Aktionen (welche Kriterien)

A rational agent chooses whichever action maximizes the expected value of
the performance measure given the percept sequence to date
Rational != omniscient: percepts(anlayıs,sezıs) may not supply all relevant information
Rational != clairvoyant(gorulmeyen seyleri gorebilen kimse): action outcomes may not
be as expected
Hence, rational != successful

2.6 Agent Type

Human Agent has eyes, ears and other organs for sensors. Hands , legs and other parts of body for effectors.
Robotic Agent camera and infrared range fingers fort he sensors and vairous motor for effectors.
Software agent has encoded bit strings as its percepts and actions.

AGENT , RICHTIG FALSCH FRAGEN.

Wenn es prinzipiell eine Lösung gibt, dann wird diese auch von einem rationalen Agenten gefunden.

A rational Agent is omniscient (german ;alwissend) (no)

A rational Agent chooses whichever action maximizes the expected value oft he performance measure .

Ein Agent versucht immer den erwarteten Gewinn zu maximieren

3.LOGIK (Ontology Engineering)

3.1 Vorteil von deklarativen Wissensrep. zur prozuderialen.
Declarative knowledge representation techniques
Advantages:
∗ increased versatility(cokyonluluk) for performing complex tasks
∗ changes can be easily incorporated (modularity)

3.2 Manche Tiere sind weder Bären noch Tiger (A(x)=Tier, B(x)/T(x) = Bär/Tiger) als Prädikatenlogikformel μ angeben .

• Examples:
– "I would do that for anyone." =⇒ ∀xA(x)
– "I wouldn't do that for anyone." =⇒ ¬∃xA(x)
– "If any man is godfearing, he is just." =⇒ ∀x(G(x) ⇒ J(x))
– "If any man is just, Aristides is just." =⇒ (∃xJ(x)) ⇒ J(a)
– "If Superman is a villain, then any man is a villain."
=⇒ V (s) ⇒ ∀xV (x)
Basic concepts of logic:
– syntax: formal structure of sentences
– semantics: truth of sentences wrt models
– entailment: necessary truth of one sentence given another
– inference: deriving sentences from other sentences
– soundness: derivations produce only entailed sentences
– completeness: derivations can produce all entailed sentences

3:3 Extrinsisch/intrinisch erklären.
Intrinsic Properties : belong to the very substance of the object, rather than to the object as a whole. When you cut a substance in half, the pieces retain the same set of properties. Like density, boiling, point, flavor, color etc.
Extrinsic Properties: Those which are not retained under subdivision like weight, length, shape etc.

3.4 Erklären Sie den Unterschied zwischen "stuff" und "thing", geben Sie beispiele
The distinction things and stuff corresponds to the following

- The category Stuff is the most general substance category, specifying no intrinsic(density, boiling degree) properties.
- The category Thing is the most general discrete object category , specifying no extrinsic(length, shape) properties.

We can therefore say a substance, or a mass noun, is a class of object that includes in its definition only intrinsic properties.
Things--: intrinsic, mass noun, substance, butter, water energy.
Stuff -- : extrinsic, count noun,

3.5 "Österreicher sind aus Europe"
A=Universal Quantifier, _E= Existential Quantifier , ^=log and)

1. Ax (Ö(x) => Europe(x)) .
2. Ex(Ö(x) ^Europe(x))
3. ExÖ(x) ^ **Ax(Ö(x) => Europe(x)) // pay attention parenthesis (klammer) (Regel ,Folie 35)**

*****Rule 3 is not requested**

3.6 FOL

1.Alle Menschen sind Sterblich .	Ax (Menschen(x) =>sterblich(x)) * klammerlara dikkat.
2.Mansche Menschen sind Raucher.	Ex(Menschen(x) ^ Raucher(x))
3.Alle Raucher sind Sterblich.	Ax (Raucher(x) =>sterblich(x)).

*** Wird Satz 3 von Satz 2 und 1 subsumiert ? NEIN. Nicht Menschlich aber Raucher sein (z.b Engel)**

*** Wenn alle Engel Raucher sind, dann gibt es einige Engel die Rauchen.**
[∃x Engel(x) ∧ ∀x (Engel(x) => Raucher(x))] => ∃x (Engel(x) ∧ Raucher(x))

(Folien 10-1 , 35.)

- In natural language, "all S are P" would normally not be asserted if it is already known that S does not hold.
- Indeed, people would not consider "all S are P" true if S is false.
  $\Longrightarrow$ "all S are P" would in this sense be translated as
  $$\exists x S(x) \wedge \forall x(S(x) \Rightarrow P(x))$$
  rather than as $\forall x(S(x) \Rightarrow P(x))$.

The Terms :

Gültig: All of is true
Erfüllbar: both of true and false.
Unerfüllbar: all of is false

Jeder kennt jemanden.
1Jeder kennt jemanden : Ax Ey (K(x,y))
2 Jeder wird von jedem gekannt: Ax Ay (K (x,y))

Wird 2 von 1 impliziert ? Ax Ey (K (x, y)) => Ax Ay (K(x,y)) is falsch. Wenn jeder jemanden kennt, kann es sein, daß jeder nur sich selbst kennt (oder genau eine andere Person und es sind > 2

Ax Ay (K(x,y)) => Vx Ey (K(x,y)) ist richtig - Wenn jeder von jedem gekannt wird, dann kennt jeder jeden, also zumindestens einen.

3.7 Partition

A disjoint exhaustive decomposition is a partition.

Partition (s,c) <-> (Disjoint(s)^ Exhaustive Decomposit (s,c))

4 PLANING

We are only concerned with classical planning environments, which are
– fully observable,
– deterministic,
– finite,
– static (change happens only when the agent acts), and
– discrete (in time, action, objects, and effects).

4.1 Aus was besteht ein STRIPS Plan (=Aktion/Precon/Effekt)

STRIPS (Fikes and Nilsson, 1971), a basic representation language of classical planners.
"STRIPS" stands for "Stanford Research Institute Problem Solver"
STRIPS insist of STRIPS Syntax and STRIPS semantic.
STRIPS Syntax:
representation of goal
represantation of actions
representation of actions
STRIPS semantic: Chapter 11 -1
http://tuwis.tuwien.ac.at/zope/_ZopeId/10589729A3PTI5hYHhE/tpp/lv/lva_html?num=184262&sem=2008S

4.2 Wie löst STRIPS das Frame Problem.

Assumption: Every literal not mentioned in the effect remains unchanged in the resulting state when the action is executed.
Avoids the representational frame problem. Solution for the planning problem:
An action sequence that, when executed in the initial state, results in a state that satisfies the goal.

Problem: Actions don't specify what happens to objects not involved in the action, but the logic framework requires that information

Frame Axioms: Inform the system about preserved relations.

Procedural solutions to the **frame problem** have been around since the earliest days of the **frame problem**. The best known of such approaches is **STRIPS** (Fikes and Nilsson, 1971).

STRIPS is rst and foremost a planning program: Given an initial state, a goal state, and a list of actions, it will plan a sequence of actions that achieves the goal state, using a planning method known as means-end reduction.
STRIPS solves the **frame problem** by assuming that if an action is not specically known to change some feature of the world, it does not. This principle is easy to represent procedurally (thus the popularity of procedural solutions to the **frame problem**).

http://209.85.135.104/search?q=cache:jypVEnTd5UYJ:www-formal.stanford.edu/leora/fp.ps+How+to+solve+the+STRIPS+frame+problem&hl=de&ct=clnk&cd=37&gl=at

4.3 Multiple Choice Fragen zu Unterschied zw. ADL und STRIP

STRIPS	ADL
Only positive literals in states: $Rich \wedge InJail$	Positive and negative literals in states: $\neg Poor \wedge \neg Free$
Closed-World Assumption: Unmentioned literals are false	Open-World Assumption Unmentioned literals are unknown
Effect $P \wedge \neg Q$ means add P and delete Q	Effect $P \wedge \neg Q$ means add P and $\neg Q$ and delete $\neg P$ and Q
Only ground literals in goals: $Rich \wedge InJail$	Quantified variables in goals: $\exists x \;\; At(P_1, x) \wedge At(P_2, x)$ is the goal of having P_1 and P_2 in the same place
Goals are conjunctions: $Rich \wedge Famous$	Goals allow conjunction and disjunction: $\neg Poor \wedge (Famous \vee Smart)$
Effects are conjunctions:	Conditional effects are allowed: $when P : E$ means E is an effect only if P is satisfied
No support for equality	Equality is built in
No support for types	Variables can have types, as in $(p : Plane)$

In ADL, the Fly action can be written as follows:

$$Action(Fly(p : Plane, from : Airport, to : Airport),$$
$$\text{PRECOND: } At(p, from) \wedge from \neq to$$
$$\text{EFFECT: } \neg At(p, from) \wedge At(p, to))$$

4:4 Was ist ein Partial Order Planning, was ist der Vorteil gegenüber Total Order Plan?
To understand what partial-order planning entails, it might be helpful to know what planning is, and then what ordered planning entails. To that end, planning is "the task of coming up with a sequence of actions that will achieve a goal
Any planning algorithm that can place two actions into a plan without specifying which comes first is called a partial-order planner.
We start with an empty plan.
• Then we consider ways of refining the plan until we come up with a complete plan that solves the problem.
• The actions in this search are not actions in the world but actions on

Partial order plan components.
1. a set of actions;
2. a set of ordering constraints;
3. a set of causal links;
4. a set of open preconditions.

The partial-order plan is a solution is the set of open conditions is empty and if there is no conflict. The empty plan contains just the Start and Finish actions.
– Start has no preconditions and has as its effect all the literals in the initial state of the planning problem.
– Finish has no effects and has as its preconditions the goal literals of the planning problem.

An ordering constraint is a pair of actions of the form , read $A \prec B$, as "A before B
Any cycle, like $A \prec B$ and $B \prec A$, represents a contradiction
$\Longrightarrow$ a ordering constraint cannot be added to the plan if it creates a cycle.

Advantages
– Partial order planning is ***sound*** and ***complete***

– Typically produces ***optimal*** solutions (plan length)
– Least commitment may lead to shorter search times

4.5 was ist das Ramification Problem, Was ist das Quality Problem
Ramification(Dallanma) Problem: Ramification problem concerns the proliferation (cogalma) of implicit consequences actions.

The Qualificiation Problem: Arise because its difficult. In the real world, to define the circumstances under which a given action is guaranteed to work.

4.6 Was sind unique-name axiome?
The following formulas are referred to as unique-action axioms.

1. For each pair of action names A and B, we assume
$$A(x_1, \ldots, x_m) \neq B(y_1, \ldots, y_n).$$
2. Two action terms with the same action name refer to the same action only if they involve all the same objects:
$$A(x_1, \ldots, x_m) = A(y_1, \ldots, y_m) \;\Leftrightarrow\; x_1 = y_1 \wedge \ldots \wedge x_m = y_m.$$

4.7 Unterschied Regression Planning und Progression Planning

Progression planning is a way from initial state to goal. (İlerleme).

Regression planning is a way from goal to initial state . (gerileme).

Forward State Space Search → Progressing Planning

Backward State Space Search→ Regression Planning.

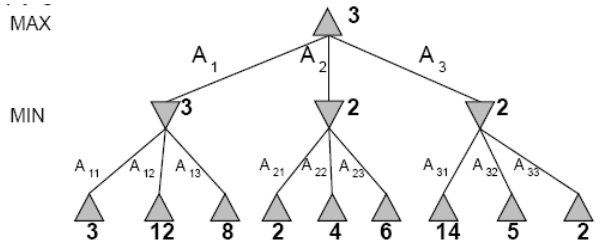
5 GAME

5.1 MINIMAX ALGORITHM
Welche eigenschaften muss ein spiel haben, damit der MINIMAX algorithmus funktioniert. ist minimax optimal? Begründung.

The types of games are perfect information, imperfect information, deterministic and change.

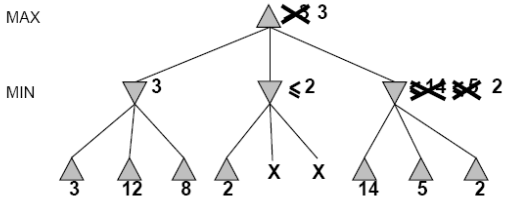
Für welche Art von Spielen macht minimax ein perfektes Spiel ?
For the minimax algorithm **perfect play for deterministic, perfect- information games** are important.
The idea of minimax algorithm is choose move to position with highest minimax value.

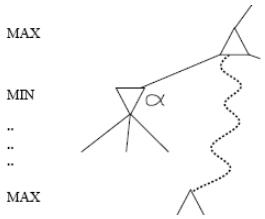
The nodes are moves by max (a normal triangle)
The nodes are moves by min(turn down)



5.2 Minimax is Optimal and Complete?
Complete: yes, if tree is finite . Chess is a specific rules for this.
Optimal : yes , against an optimal opponent. Minimax is an optimal method for selecting a move from a given search tree.

5.3 Alpha- Beta Pruning





α is the best value (to MAX) found so far off the current path

If V is worse than α, MAX will avoid it $\Rightarrow$ prune that branch

Define β similarly for MIN

If alfa is better than n for player , we will newer get to n in play.

5.4 Nenne die 4 Typen von Spielen und gib jeweils ein Beispiel an.

The types of games : perfect information -- chess
Imperfect information---- battleship
Deterministic --- checkers, chess
Non deterministic--- backgammon

5.6 Wie ist die Time complexity und Space Complexity von Minimax?

m, max depth of the tree.b , legal moves at cach point.

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

5.7 Minimax baum aufzeichnen mit Ergebnisse der Eval Funktion in jeden schritt.

Eval Function: instead of utility. Evaluation function that estimate desirability of position .
Evaluation function allow us to approximate the true utility of state without doing a complete search.

6. UNCERTAINTY

6.1 Wie lautet das Bayerische Theorem?

- Reconsider two instances of the product rule:

$$\mathrm{P}(V_i, V_j) = \mathrm{P}(V_i|V_j)\mathrm{P}(V_j)$$
$$\mathrm{P}(V_i, V_j) = \mathrm{P}(V_j|V_i)\mathrm{P}(V_i)$$

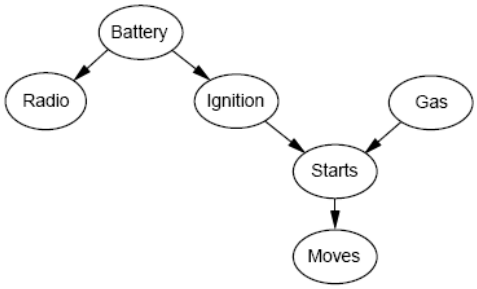
- Equating the two right-hand sides and dividing by $\mathrm{P}(V_j)$, we get

$$\boxed{\mathrm{P}(V_i|V_j) = \frac{\mathrm{P}(V_j|V_i)\mathrm{P}(V_i)}{\mathrm{P}(V_j)}}$$

  – This equation is known as Bayes' rule (after Ref. Thomas Bayes, 1702–1761).

6.2 Bayesian Network

- Each node is conditionally independent of its non-descendants, given its parents.
- Hence, $\mathrm{P}(V|\mathbf{W}, Parents(V)) = \mathrm{P}(V|Parents(V))$, for any set $\mathbf{W}$ of nodes which are neither parents nor descendants of V.



This graph encodes, e.g., the following independence relations:

$$
\begin{aligned}
P(\text{Starts}|\text{Battery}, \text{Radio}, \text{Ignition}, \text{Gas}) &= P(\text{Starts}|\text{Ignition}, \text{Gas}) \\
P(\text{Gas}|\text{Battery}, \text{Radio}, \text{Ignition}) &= P(\text{Gas}) \\
P(\text{Ignition}|\text{Radio}, \text{Gas}, \text{Battery}) &= P(\text{Ignition}|\text{Battery}) \\
P(\text{Battery}|\text{Gas}) &= P(\text{Battery}) \\
P(\text{Radio}|\text{Gas}, \text{Battery}) &= P(\text{Radio}|\text{Battery})
\end{aligned}
$$