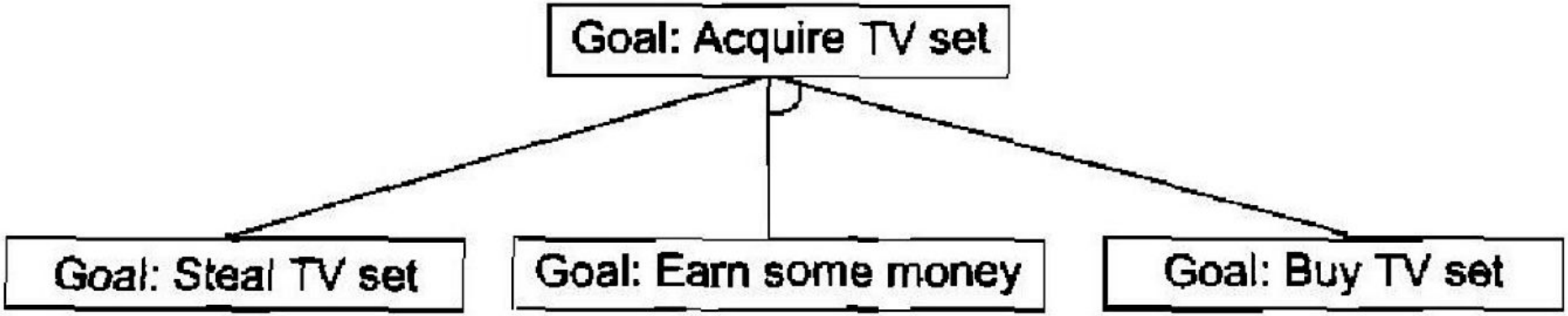
AO*



An example of an AND-OR graph
(which also happens to be an AND-OR tree).

AND-OR Graphs

- AND-OR graph is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems.

AND-OR Graphs

- AND-OR graph is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems.
- AND arc may point to any number of successor nodes, all of which must be solved.

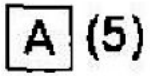
AND-OR Graphs

- AND-OR graph is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems.
- AND arc may point to any number of successor nodes, all of which must be solved.
- OR arch indicate a variety of ways in which the original problem might be solved.

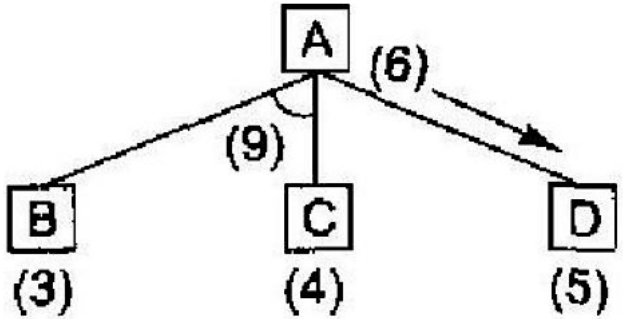
AND-OR Graphs

- AND-OR graph is useful for representing the solution of problems that can be solved by decomposing them into a set of smaller problems.
- AND arc may point to any number of successor nodes, all of which must be solved.
- OR arch indicate a variety of ways in which the original problem might be solved.
- To find solutions in an AND-OR graph, we need an algorithm with the ability to handle the arcs appropriately.

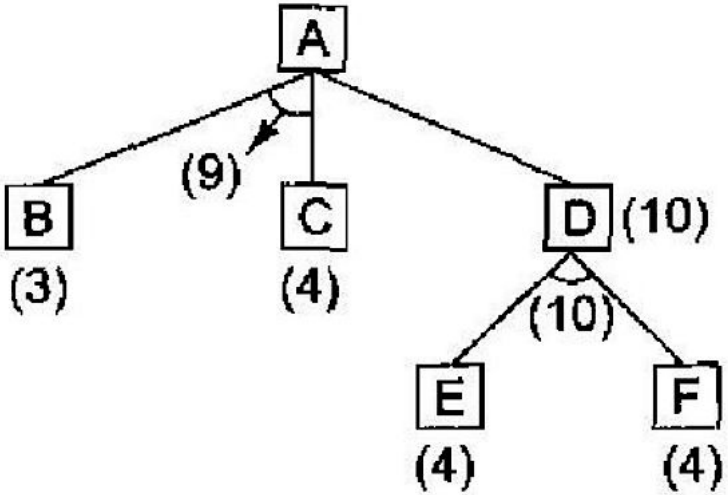
Before step 1



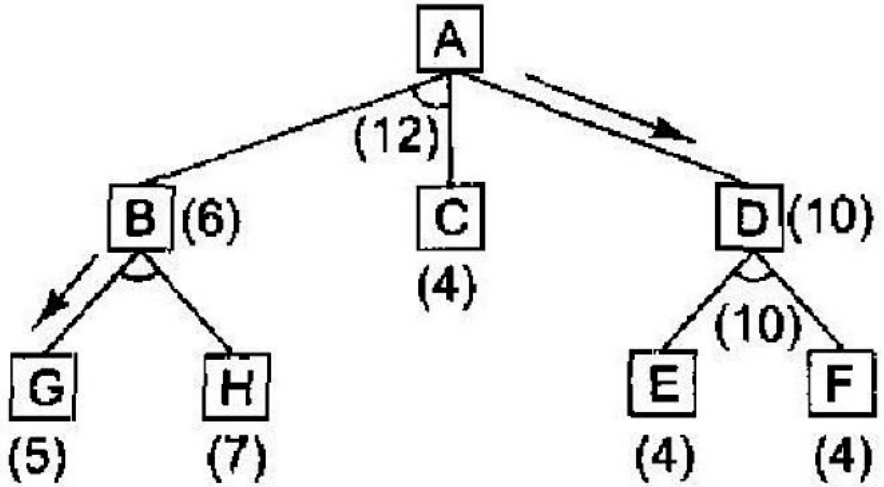
Before step 2



Before step 3



Before step 4



Problem Reduction

- Initialize the graph to the starting node.

Problem Reduction

- Initialize the graph to the starting node.
- Loop until the starting node is labeled **SOLVED** or until its cost goes above **FUTILITY**.

Problem Reduction

- Initialize the graph to the starting node.
- Loop until the starting node is labeled **SOLVED** or until its cost goes above **FUTILITY**.
- Traverse the graph, starting at the initial node and following the current best path, and accumulate the set of nodes that are on that path and have not yet been expanded or labeled as solved.

Problem Reduction

- Initialize the graph to the starting node.
- Loop until the starting node is labeled **SOLVED** or until its cost goes above **FUTILITY**.
- Traverse the graph, starting at the initial node and following the current best path, and accumulate the set of nodes that are on that path and have not yet been expanded or labeled as solved.
- Pick one of these **unexpanded nodes** and expand it.

Problem Reduction

- Initialize the graph to the starting node.
- Loop until the starting node is labeled **SOLVED** or until its cost goes above **FUTILITY**.
- Traverse the graph, starting at the initial node and following the current best path, and accumulate the set of nodes that are on that path and have not yet been expanded or labeled as solved.
- Pick one of these **unexpanded nodes** and expand it.
- If there are no successors, assign **FUTILITY** as the value of this node. Otherwise, add its successors for future expansion.

Problem Reduction

- If of any node is 0 , mark that node as **SOLVED**.

Problem Reduction

- If of any node is 0 , mark that node as **SOLVED**.
- Compute the estimate cost of the newly expanded node get the revised cost estimates. Propagate this change backward through the graph.

Problem Reduction

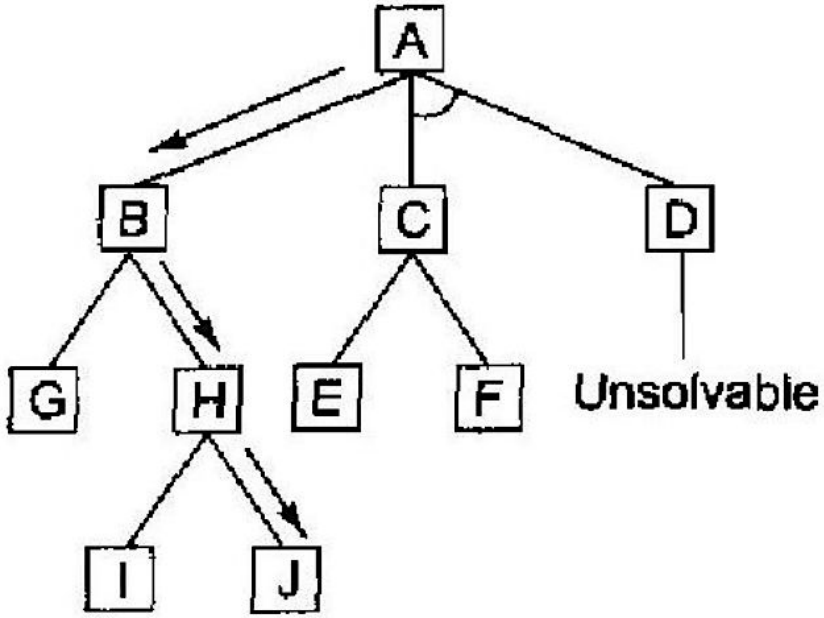
- If of any node is 0 , mark that node as **SOLVED**.
- Compute the estimate cost of the newly expanded node get the revised cost estimates. Propagate this change backward through the graph.
- If any node contains a successor arc whose descendants are all solved, label the node itself as SOLVED.

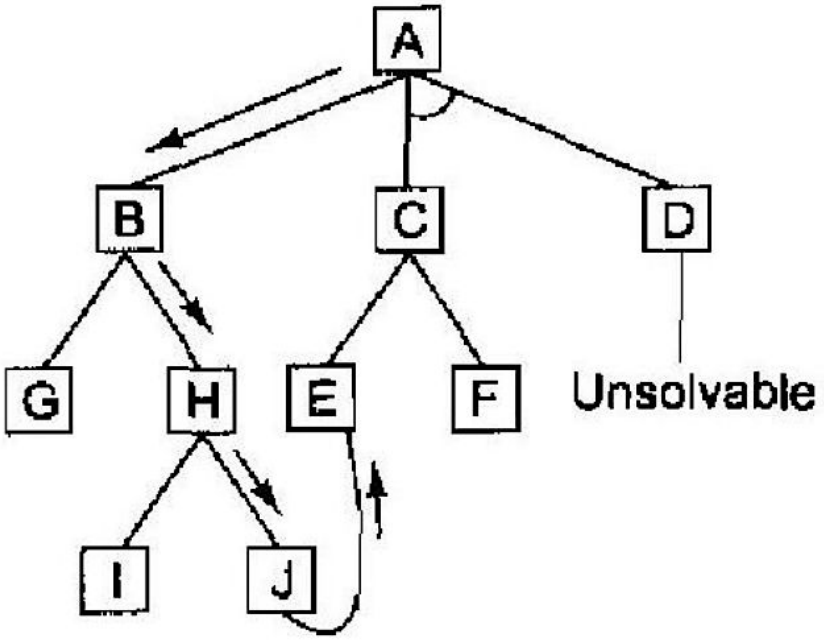
Problem Reduction

- If of any node is 0 , mark that node as **SOLVED**.
- Compute the estimate cost of the newly expanded node get the revised cost estimates. Propagate this change backward through the graph.
- If any node contains a successor arc whose descendants are all solved, label the node itself as SOLVED.
- At each node decide which of its successor arcs is the most promising and mark it as part of the current best path.

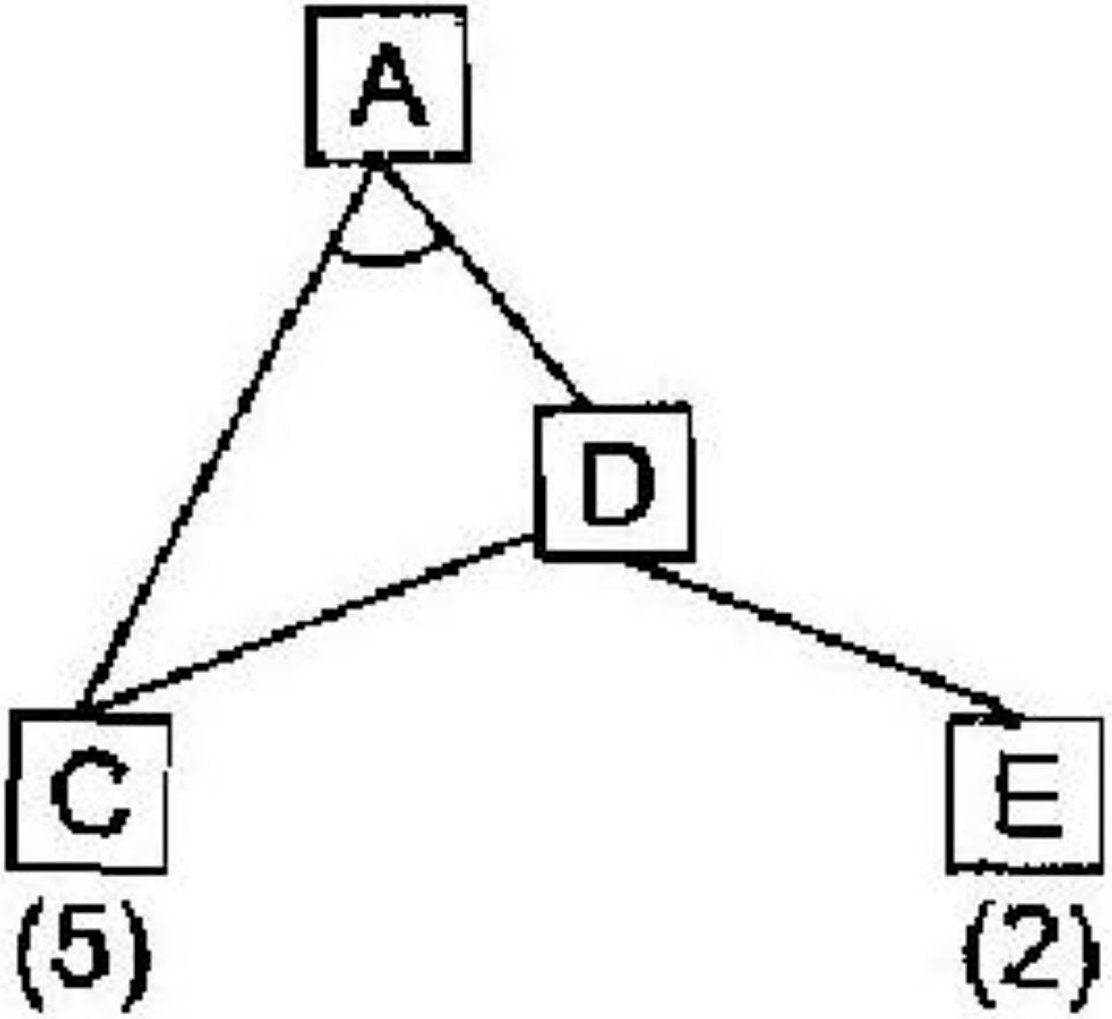
Problem Reduction

- If of any node is 0 , mark that node as **SOLVED**.
- Compute the estimate cost of the newly expanded node get the revised cost estimates. Propagate this change backward through the graph.
- If any node contains a successor arc whose descendants are all solved, label the node itself as SOLVED.
- At each node decide which of its successor arcs is the most promising and mark it as part of the current best path.
- Note that the propagation of revised cost estimates back up the tree was not necessary in the best-first search algorithm because only unexpanded nodes were examined.

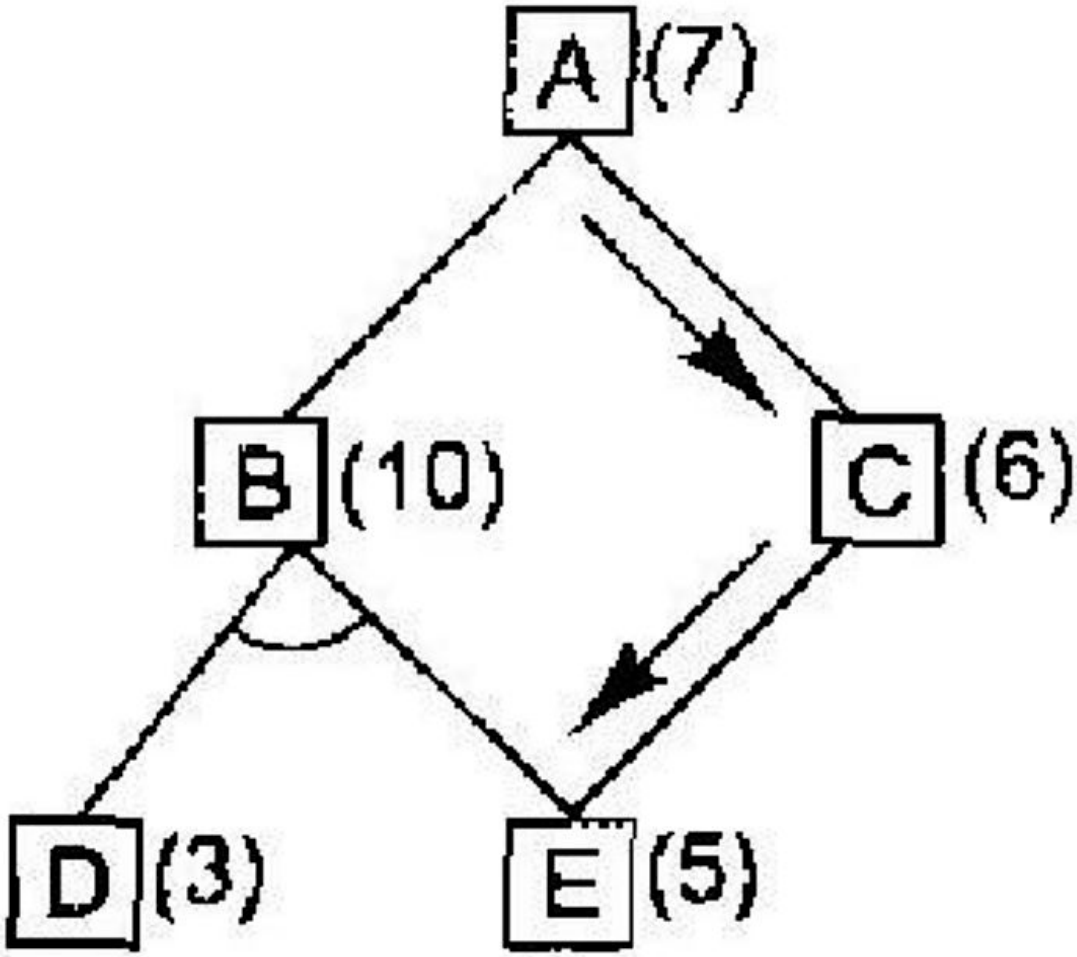




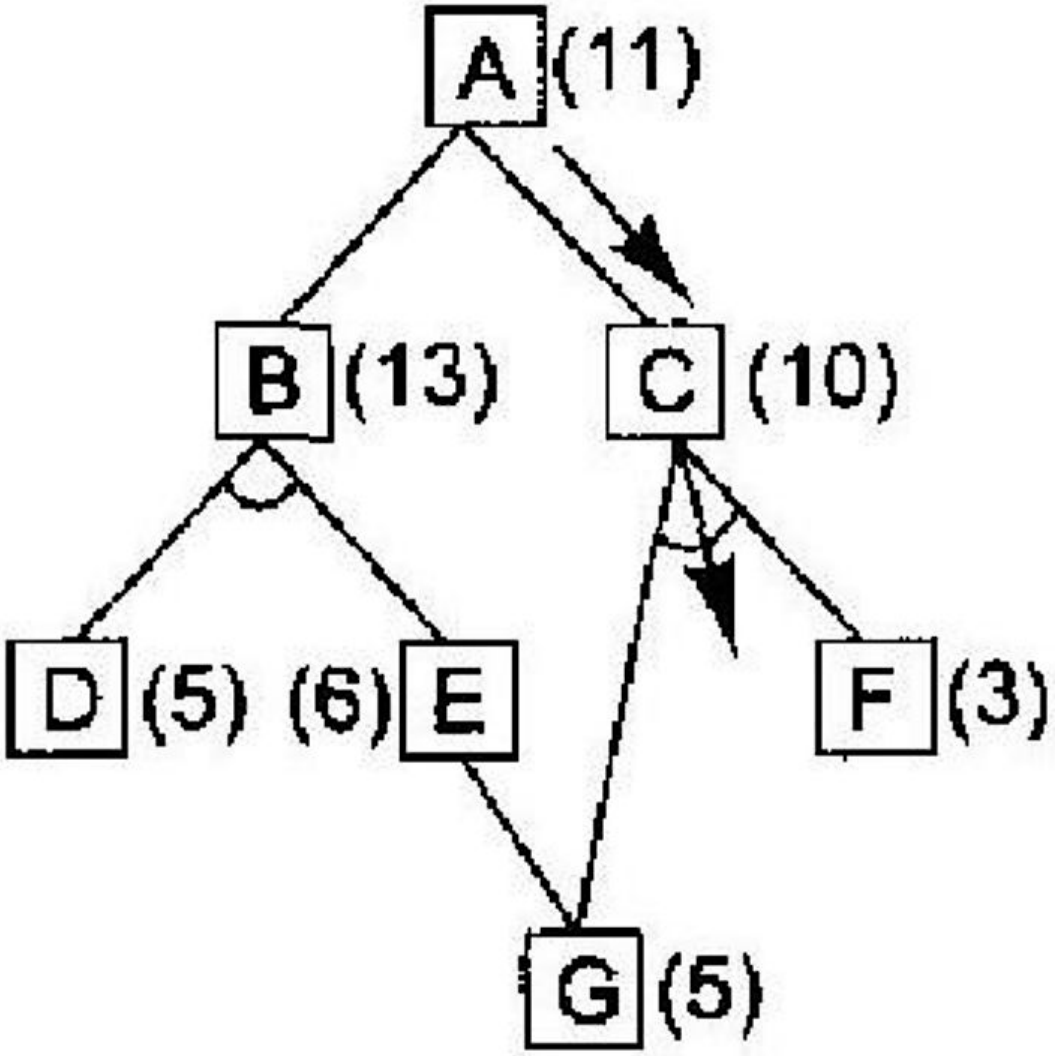
A Longer Path May Be Better



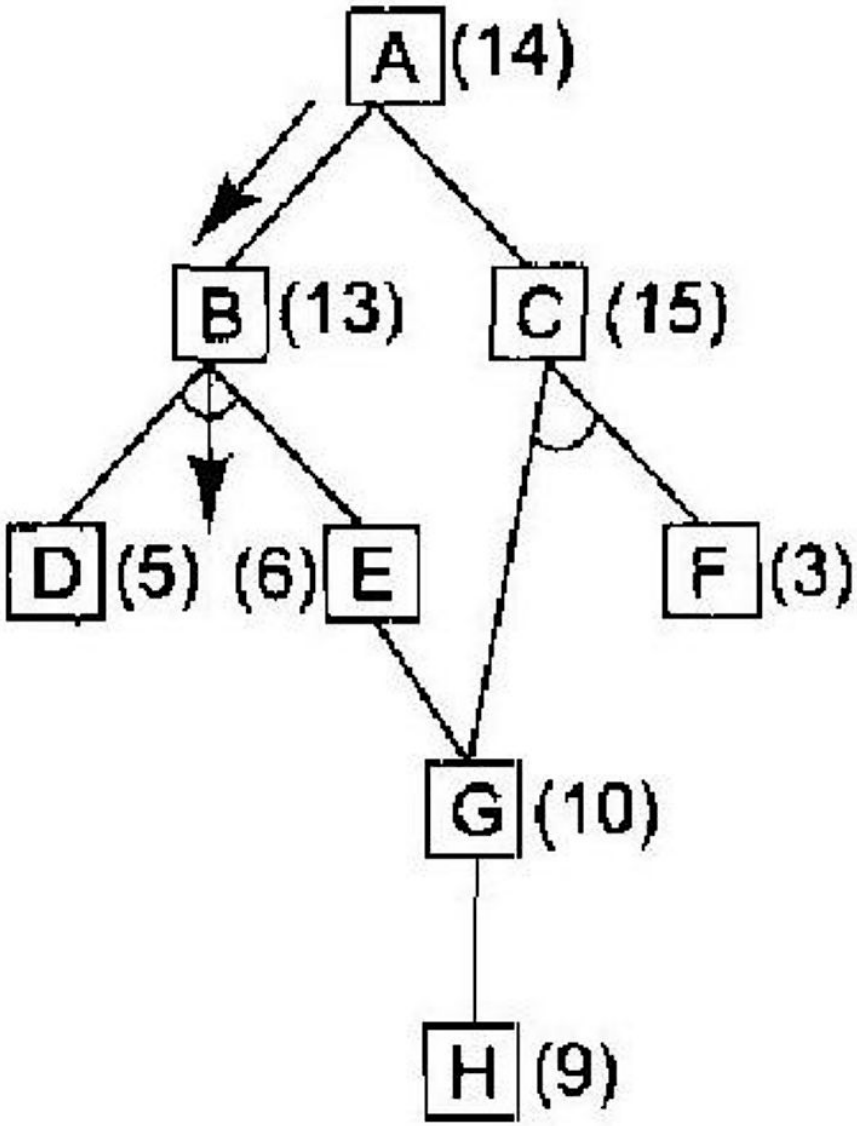
Interacting Subgoals



An Unnecessary Backward Propagation changes and is propagated back to B. Time wasted in expanding D.



(a) A Necessary Backward Propagation



(b) A Necessary Backward Propagation

Introduction to Knowledge Representation

Knowledge

A Knowledge is simply *a*ssured belief or facts.

For example:

- *It is raining.*
- *It is sunny.*
- *It is windy.*
- *If it is raining, then it is not sunny.*

In order to solve the complex problems a large amount of **knowledge** and suitable mechanisms for **representing** and manipulating all that knowledge.

Knowledge

A Knowledge is simply *a*ssured belief or facts.

For example:

- *It is raining.*
- *It is sunny.*
- *It is windy.*
- *If it is raining, then it is not sunny.*

In order to solve the complex problems a large amount of knowledge and suitable mechanisms for representing and manipulating all that knowledge.

So, how should an AI agent store and manipulate knowledge like this?

Entities in Knowledge Representation

There are two different kinds of entities, we are dealing with.

- **Facts**: Truth in some relevant world. Things we want to represent.
- **Representation of facts** in some chosen formalism. Things we will actually be able to manipulate. E.g., Logic, Rules, Frames, and Semantic Net

Knowledge Representation Language

A ***knowledge representation language*** is defined by two aspects:

1. **Syntax:** The configurations of the components of the language that constitute a valid sentences.
2. **Semantics:** The facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

Knowledge Representation Language

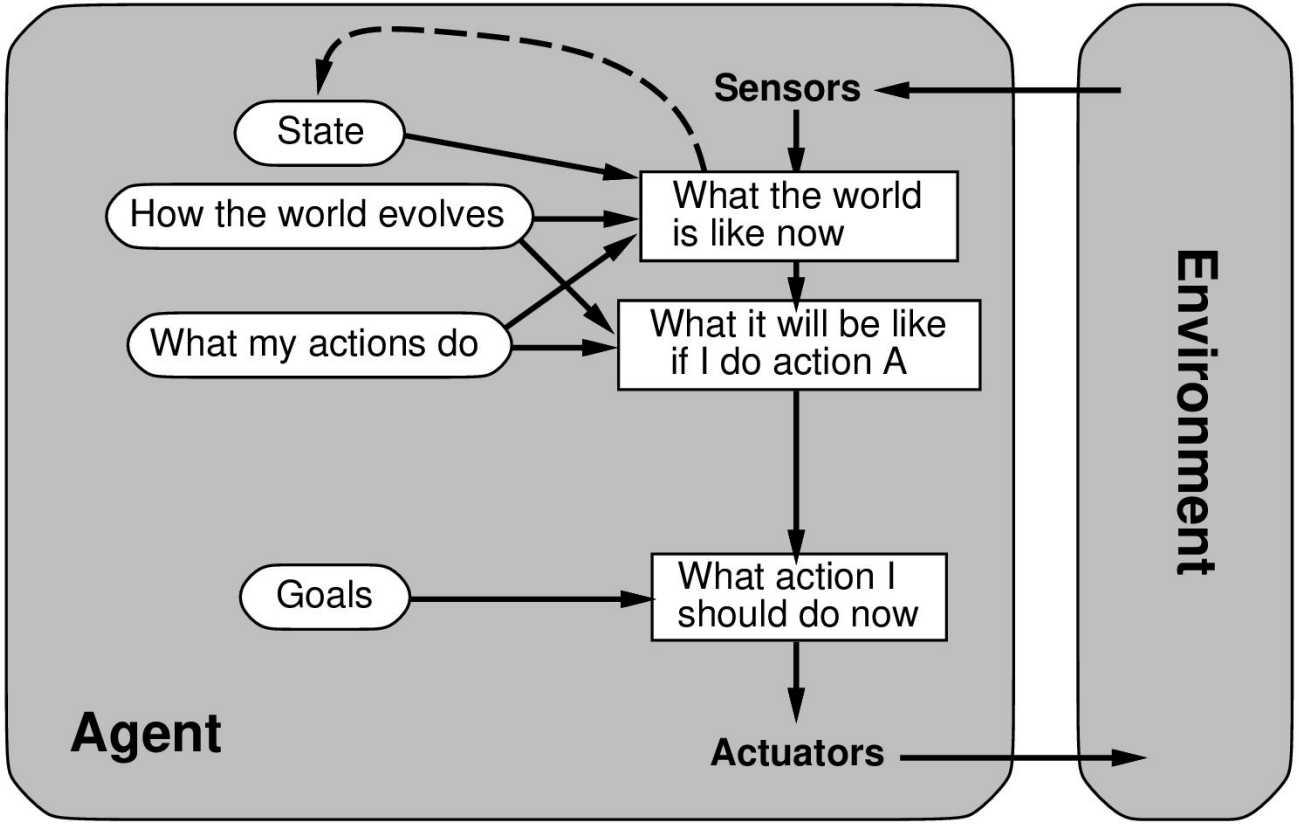
A ***knowledge representation language*** is defined by two aspects:

1. **Syntax:** The configurations of the components of the language that constitute a valid sentences.
2. **Semantics:** The facts in the world the sentences refer to, and hence the statement about the world that each sentence makes.

Suppose arithmetic language with '*x*', '≥' and '*y*' as its *components* (or symbols or words).

- The *syntax* says that **'*x* ≥ *y*'** is a valid sentence in the language, but **'≥ ≥ x y'** is not.
- The *semantics* say that **'*x* ≥ *y*'** is false if *y* is bigger than *x*, and true otherwise.

Remember our goal-based agent



Abstraction Levels

We can describe every reasoning agent (natural or not) at two different abstraction levels :

1. Knowledge level: what the agent knows and what the agent's goals are
2. Symbol (or implementation) level: what symbols the agent manipulates and how

Abstraction Levels

We can describe every reasoning agent (natural or not) at two different abstraction levels :

1. Knowledge level: what the agent knows and what the agent's goals are
2. Symbol (or implementation) level: what symbols the agent manipulates and how

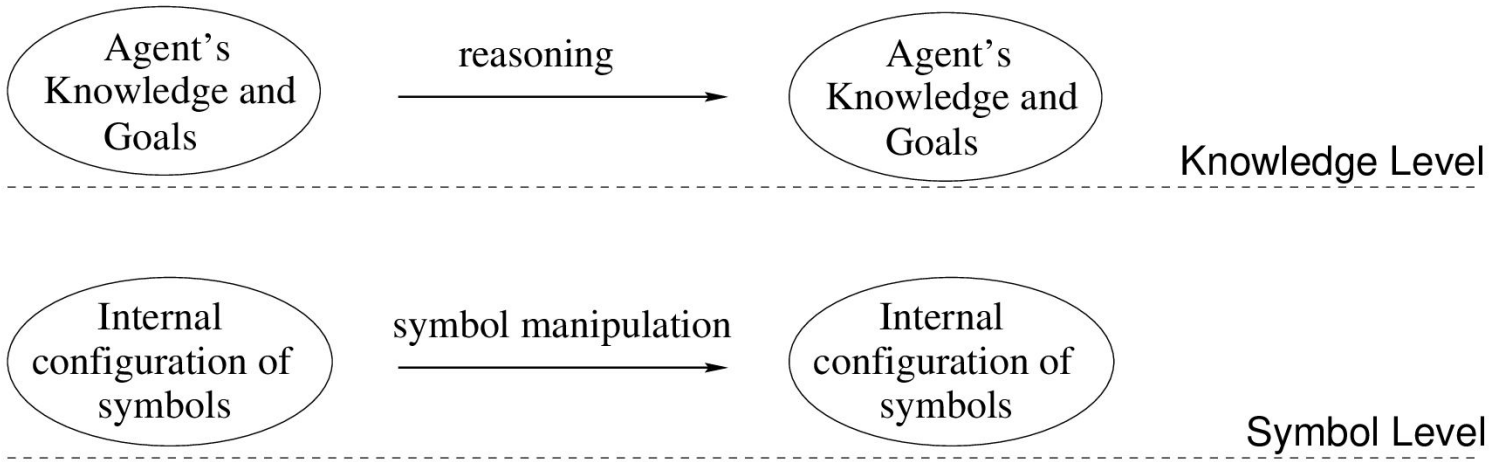
Agents can be viewed at

- the knowledge level
  i.e., what they know and what they can infer, regardless of how implemented
- or at the implementation level
  i.e., data structures to store knowledge and algorithms to manipulate them

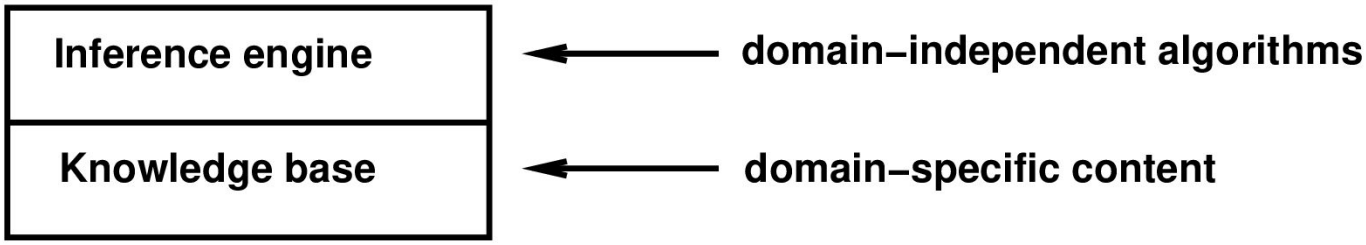
Abstraction Levels

We can describe every reasoning agent (natural or not) at two different abstraction levels :

1. Knowledge level: what the agent knows and what the agent's goals are
2. Symbol (or implementation) level: what symbols the agent manipulates and how

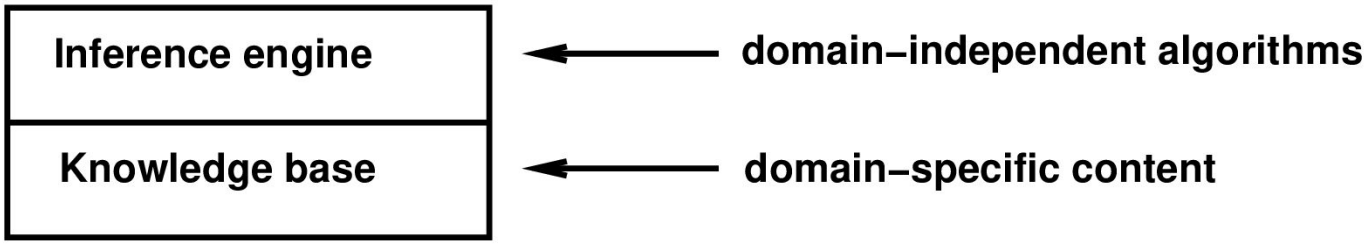


Knowledge bases



Knowledge base (KB) = set of sentences in a formal language

Knowledge bases



Knowledge base (KB) = set of sentences in a formal language

Declarative approach to building an agent (or other system):

- TELL it what it needs to know
- Then it can ASK itself what to do
- Answers are consequences of the KB

Logics

- **Logic** is the study of correct reasoning.
- Logical **reasoning** is concerned with arriving at a conclusion using Inferences (premises to conclusion).
- Example:
  - All humans are mortal.
  - All Greeks are humans.
  - All Greeks are mortal.
    - premises and conclusion are true

Logics

A logic is a triple $\langle \mathcal{L}, \mathcal{S}, \mathcal{R} \rangle$ where

- $\mathcal{L}$, the logic's language, is a class of sentences described by a formal grammar
- $\mathcal{S}$, the logic's semantics is a formal specification of how to assign *meaning* in the "real world" to the elements of $\mathcal{L}$
- $\mathcal{R}$, the logic's inference system, is a set of formal derivation *rules* over $\mathcal{L}$

Logics

A logic is a triple $\langle \mathcal{L}, \mathcal{S}, \mathcal{R} \rangle$ where

- $\mathcal{L}$, the logic's language, is a class of sentences described by a formal grammar
- $\mathcal{S}$, the logic's semantics is a formal specification of how to assign *meaning* in the "real world" to the elements of $\mathcal{L}$
- $\mathcal{R}$, the logic's inference system, is a set of formal derivation *rules* over $\mathcal{L}$

There are several logics: propositional, first-order, higher-order, modal, temporal, intuitionistic, linear, equational, non-monotonic, fuzzy, ...

We will concentrate on propositional logic and first-order logic

Propositional Logic

Each sentence is made of

- propositional variables ($A, B, \ldots, P, Q, \ldots$)
- logical constants ($\mathbf{True}, \mathbf{False}$)
- logical connectives ($\wedge, \vee, \Rightarrow, \ldots$)

Every propositional variable stands for a basic fact

Examples: *I'm hungry, Apples are red, Joe and Jill are married*

Propositional Logic

Each sentence is made of

- propositional variables ($A, B, \ldots, P, Q, \ldots$)
- logical constants ($\mathbf{True}, \mathbf{False}$)
- logical connectives ($\wedge, \vee, \Rightarrow, \ldots$)

Model: an assignment of (True/False) values to each of the variables. If the knowledge base is built from n variables, there are 2^n possible models.

Propositional Logic

Ontological Commitments

Propositional Logic is about facts in the world that are either true or false, nothing else

Semantics of Propositional Logic

Since each propositional variable stands for a fact about the world, its meaning ranges over the Boolean values $\{true, false\}$

Propositional Logic

Ontological Commitments

Propositional Logic is about facts in the world that are either true or false, nothing else

Semantics of Propositional Logic

Since each propositional variable stands for a fact about the world, its meaning ranges over the Boolean values $\{true, false\}$

Note: Do note confuse

- $true$, $false$, which are values (i.e., semantical entities) here with
- **True**, **False**, which are logical constants (i.e., symbols of the language)

Propositional Logic

The Language

- Each propositional variable ($A, B, \ldots, P, Q, \ldots$) is a sentence
- Each logical constant ($\mathbf{True}, \mathbf{False}$) is a sentence
- If φ and ψ are sentences, all of the following are also sentences

$$(\varphi) \qquad \neg\varphi \qquad \varphi \wedge \psi \qquad \varphi \vee \psi \qquad \varphi \Rightarrow \psi \qquad \varphi \Leftrightarrow \psi$$

The Language of Propositional Logic

More formally, it is the language generated by the following grammar

Symbols:

- Propositional variables: $A, B, \ldots, P, Q, \ldots$
- Logical constants:

$\mathbf{True}$	(true)	$\wedge$	(and)	$\Rightarrow$	(implies)	$\neg$	(not)
$\mathbf{False}$	(false)	$\vee$	(or)	$\Leftrightarrow$	(equivalent)		

Grammar Rules:

$$
\begin{array}{lcl}
\mathit{Sentence} & ::= & \mathit{AtomicS} \mid \mathit{ComplexS} \\
\mathit{AtomicS} & ::= & \mathbf{True} \mid \mathbf{False} \mid A \mid B \mid \ldots \mid P \mid Q \mid \ldots \\
\mathit{ComplexS} & ::= & (\mathit{Sentence}) \mid \mathit{Sentence}\ \mathit{Connective}\ \mathit{Sentence} \mid \neg \mathit{Sentence} \\
\mathit{Connective} & ::= & \wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow
\end{array}
$$

Semantics of Propositional Logic

The meaning of **True** is always *true*
The meaning of **False** is always *false*

The meaning of the other sentences depends on the meaning of the propositional variables

- Base cases: truth tables

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Properties of Logical Connectives

- $\wedge$ and $\vee$ are commutative

$$\varphi_1 \wedge \varphi_2 \equiv \varphi_2 \wedge \varphi_1$$
$$\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$$

Properties of Logical Connectives

- $\wedge$ and $\vee$ are commutative

$$\varphi_1 \wedge \varphi_2 \equiv \varphi_2 \wedge \varphi_1$$
$$\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$$

- $\wedge$ and $\vee$ are *associative*

$$\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$$
$$\varphi_1 \vee (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \vee \varphi_3$$

Properties of Logical Connectives

- $\wedge$ and $\vee$ are commutative

$$\varphi_1 \wedge \varphi_2 \equiv \varphi_2 \wedge \varphi_1$$
$$\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$$

- $\wedge$ and $\vee$ are *associative*

$$\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$$
$$\varphi_1 \vee (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \vee \varphi_3$$

- $\wedge$ and $\vee$ are mutually *distributive*

$$\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$$
$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$$

Properties of Logical Connectives

- $\wedge$ and $\vee$ are commutative

$$\varphi_1 \wedge \varphi_2 \equiv \varphi_2 \wedge \varphi_1$$
$$\varphi_1 \vee \varphi_2 \equiv \varphi_2 \vee \varphi_1$$

- $\wedge$ and $\vee$ are *associative*

$$\varphi_1 \wedge (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \wedge \varphi_3$$
$$\varphi_1 \vee (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \vee \varphi_3$$

- $\wedge$ and $\vee$ are mutually *distributive*

$$\varphi_1 \wedge (\varphi_2 \vee \varphi_3) \equiv (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$$
$$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \equiv (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$$

- $\wedge$ and $\vee$ are related by $\neg$ (DeMorgan's Laws)

$$\neg(\varphi_1 \wedge \varphi_2) \equiv \neg\varphi_1 \vee \neg\varphi_2$$
$$\neg(\varphi_1 \vee \varphi_2) \equiv \neg\varphi_1 \wedge \neg\varphi_2$$

Normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF—universal)

$conjunction$ of $\underbrace{disjunctions \text{ of } literals}_{clauses}$

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

Conjunctive Normal Form (CNF—universal)

$conjunction$ of $\underbrace{disjunctions \text{ of } literals}_{clauses}$

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Disjunctive Normal Form (DNF—universal)

$disjunction$ of $\underbrace{conjunctions \text{ of } literals}_{terms}$

E.g., $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$

Normal forms

Other approaches to inference use syntactic operations on sentences, often expressed in standardized forms

conjunction = and
disjunction = or

Conjunctive Normal Form (CNF—universal)

$conjunction$ of $\underbrace{disjunctions \text{ of } literals}_{clauses}$

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Disjunctive Normal Form (DNF—universal)

$disjunction$ of $\underbrace{conjunctions \text{ of } literals}_{terms}$

E.g., $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$

Horn Form (restricted)

$conjunction$ of $Horn\ clauses$ (clauses with ≤ 1 positive literal)

E.g., $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Often written as set of implications:

$B \Rightarrow A$ and $(C \wedge D) \Rightarrow B$

Entailment in Propositional Logic

Given

- a set Γ of sentences and
- a sentence φ,

we write

$$\Gamma \models \varphi$$

iff every interpretation that makes all sentences in Γ true makes φ also true

$\Gamma \models \varphi$ is read as "Γ entails φ" or "φ logically follows from Γ"

Entailment in Propositional Logic

Examples

$$\{A, A \Rightarrow B\} \quad \models \quad B$$

	A	B	$A \Rightarrow B$
1.	*false*	*false*	*true*
2.	*false*	*true*	*true*
3.	*true*	*false*	*false*
4.	*true*	*true*	*true*

Entailment in Propositional Logic

Examples

$$\{A, A \Rightarrow B\} \models B$$
$$\{A\} \models A \vee B$$

	A	B	$A \Rightarrow B$	$A \vee B$
1.	*false*	*false*	*true*	*false*
2.	*false*	*true*	*true*	*true*
3.	*true*	*false*	*false*	*true*
4.	*true*	*true*	*true*	*true*

Entailment in Propositional Logic

Examples

$$\{A\} \not\models A \wedge B$$

$$\{A \vee \neg A\} \not\models A$$

	A	B	$A \Rightarrow B$	$A \vee B$	$A \wedge B$	$A \vee \neg A$
1.	*false*	*false*	*true*	*false*	*false*	*true*
2.	*false*	*true*	*true*	*true*	*false*	*true*
3.	*true*	*false*	*false*	*true*	*false*	*true*
4.	*true*	*true*	*true*	*true*	*true*	*true*

Validity and Satisfiability

A sentence is valid if it is true in all models

e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity and Satisfiability

A sentence is valid if it is true in all models

e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

Example: $A \Rightarrow (A \vee B)$ is valid and $A \models (A \vee B)$

	A	B	$A \vee B$	$A \Rightarrow (A \vee B)$
1.	*false*	*false*	*false*	*true*
2.	*false*	*true*	*true*	*true*
3.	*true*	*false*	*true*	*true*
4.	*true*	*true*	*true*	*true*

Validity and Satisfiability

A sentence is valid if it is true in all models

e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

Validity and Satisfiability

A sentence is valid if it is true in all models

e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in no models

e.g., $A \wedge \neg A$

Validity and Satisfiability

A sentence is valid if it is true in all models

e.g., $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in no models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg \alpha)$ is unsatisfiable

Logical Equivalence

Two sentences φ_1 and φ_2 are logically equivalent, written

$$\varphi_1 \equiv \varphi_2$$

if $\varphi_1 \models \varphi_2$ and $\varphi_2 \models \varphi_1$

Note:

- $\varphi_1 \equiv \varphi_2$ if and only if every interpretation assigns the same Boolean value to φ_1 and φ_2