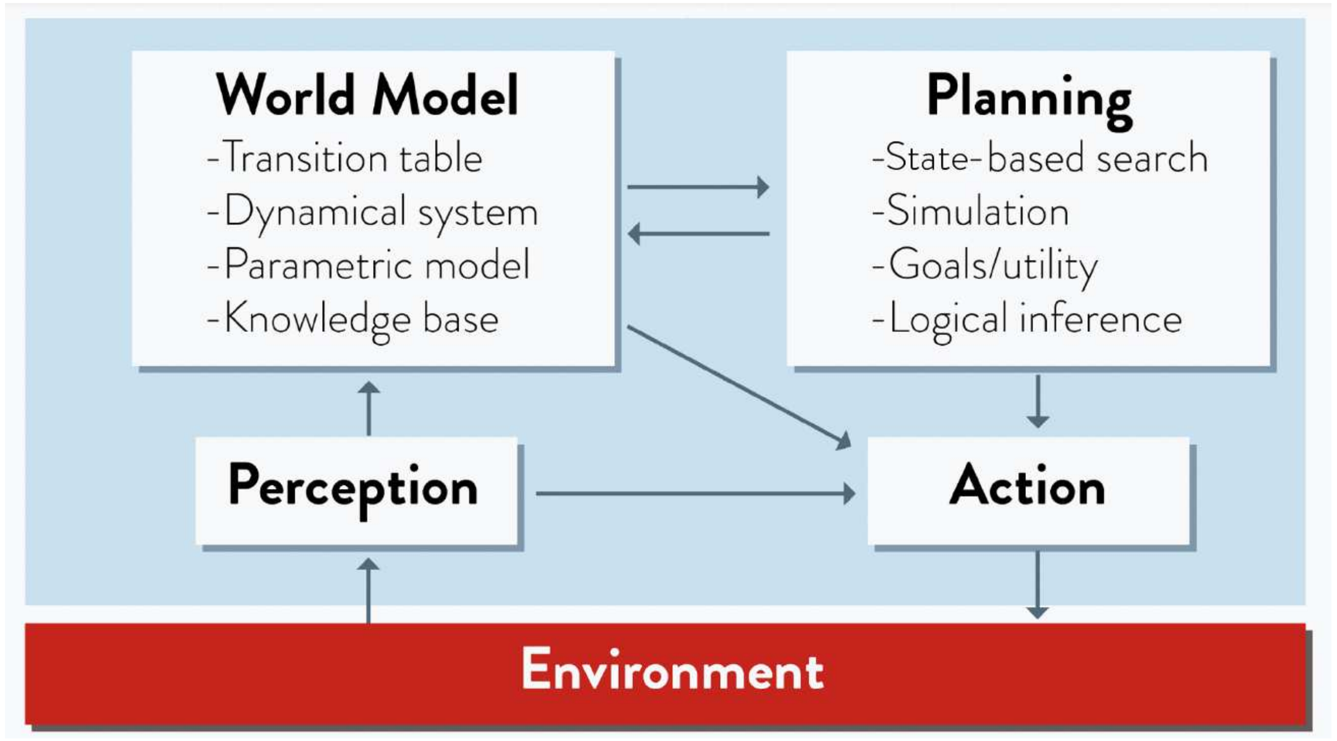
COMP3411/9814: Artificial Intelligence

Propositions and Inference

Lecture Overview

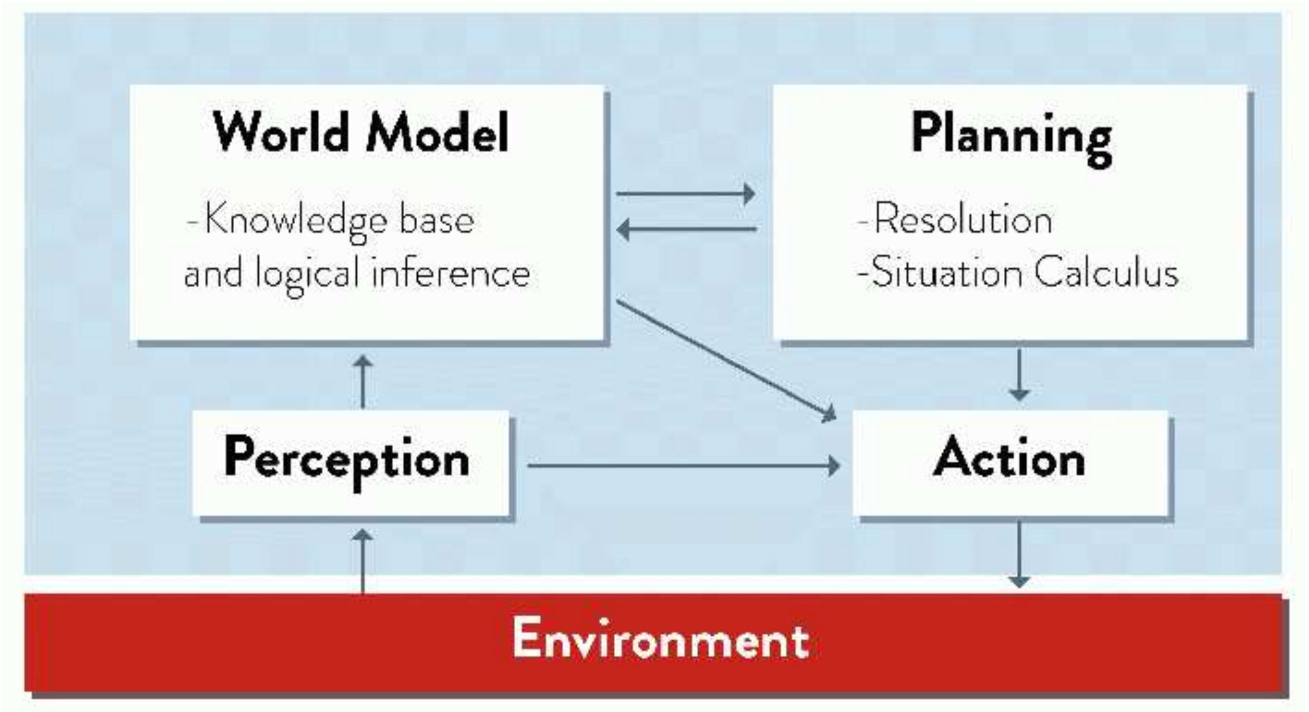
- Logic in general – models and entailment
- Propositional Logic
- Equivalence, Validity, Satisfiability
- Inference Rules and Theorem Proving
- Resolution and Conjunctive Normal Form
- Forward and Backward Chaining
- First Order Logic
- Universal and Existential Quantifiers
- Situation Calculus

Models and Planning



We previously used a transition table for our World Model, with Planning done by State-Based Search (BFS, DFS, UCS, IDS, Greedy, A*, etc.)

Models and Planning



Some environments instead require a Knowledge Base of facts and a set of Logical Inference Rules to reason about those facts.

Knowledge bases

A Knowledge Base is a set of sentences in a formal language. It takes a

- Declarative approach to building an agent (or other system):
  - Tell the system what it needs to know, then it can Ask itself what it needs to do
- Answers should follow from the KB.

Knowledge Based Agent

The agent must be able to:

- represent states, actions, etc.
- incorporate new percepts
- update internal representations of the world
- deduce hidden properties of the world
- determine appropriate actions

Propositions

- An interpretation is an assignment of values to all variables
- A model is an interpretation that satisfies the constraints.
- Often we don’t want to just find a model, but want to know what is true in all models.
- A proposition is statement that is true or false in each interpretation.

Propositions

- Propositions are entities (facts or non-facts) that can be true or false
- Expressed using ordinary declarative sentences (not questions)
  - “The sky is blue” expresses the proposition that the sky is blue
  - All humans have 2 eyes Jane has 2 eyes Therefore Jane is human

(here and now). Is this proposition true?

Propositions

Examples

- “Socrates is bald” (assumes ‘Socrates’, ‘bald’ are well defined)
- “The car is red” (requires ‘the car’ to be identified)
- “Socrates is bald and the car is red” (complex proposition)

- In Propositional Logic, use single letters to represent propositions, a scheme of abbreviation,
  - e.g. *P*: Socrates is bald

Why propositions?

- Specifying logical formulae is often more natural than filling in tables
- It is easier to check correctness and debug formulae than tables
- We need a language for asking queries
  - what follows in all models
  - may be more complicated than asking for the value of a variable
- It is easy to incrementally add formulae
- It can be extended to infinitely many variables with infinite domains (using logical quantification)

Logical Arguments

- An argument relates a set of premises to a conclusion
  - valid if the conclusion necessarily follows from the premises
    - All humans have 2 eyes
    - Jane is a human
    - Therefore Jane has 2 eyes

    - All humans have 4 eyes
    - Jane is a human
    - Therefore Jane has 4 eyes
- Both are (logically) correct valid arguments
- Which statements are true/false?

Logical Arguments

- An argument relates a set of premises to a conclusion
  - invalid if the conclusion can be false when the premises are all true
    - All humans have 2 eyes
    - Jane has 2 eyes
    - Therefore Jane is human

    - No human has 4 eyes
    - Jane has 2 eyes
    - Therefore Jane is not human
- Both are (logically) correct valid arguments
- Which statements are true/false?

Logic in general

Logics are formal languages for representing information such that conclusions can be drawn.

Syntax defines the sentences in the language.

Semantics define the “meaning” of sentences; i.e. define truth of a sentence in a world.

Logic in general

Logics are formal languages for representing information such that conclusions can be drawn.

Syntax defines the sentences in the language.

Semantics define the “meaning” of sentences; i.e. define truth of a sentence in a world.

- For example, the language of arithmetic:

  $x+2 \geq y$ is a sentence; $x2+y>$ is not a sentence

  $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y

  $x+2 \geq y$ is true in a world where $x=7,\ y=1$
  $x+2 \geq y$ is false in a world where $x=0,\ y=6$

Simple language: propositional definite clauses

- An atom is a symbol starting with a lower case letter
- A body is an atom or is of the form b1∧b2 where b1 and b2 are bodies.
- A definite clause is an atom or is a rule of the form

  $$h \leftarrow b$$

  where h is an atom and b is a body.
- A knowledge base is a set of definite clauses

From English to Propositional Logic

- "It is not the case that the sky is blue": $\neg B$ (alternatively "the sky is not blue")
- The sky is blue and the grass is green": $B \wedge G$
- "Either the sky is blue or the grass is green": $B \vee G$
- If the sky is blue, then the grass is not green": $B \rightarrow \neg G$
- "The sky is blue if and only if the grass is green": $B \leftrightarrow G$
- If the sky is blue, then if the grass is not green, the plants will not grow": $B \rightarrow (\neg G \rightarrow \neg P)$

Propositional Logic: Syntax

Propositional logic is the simplest logic

- illustrates basic ideas

The proposition symbols P_1, P_2 etc are sentences.

If S is a sentence, $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)

If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

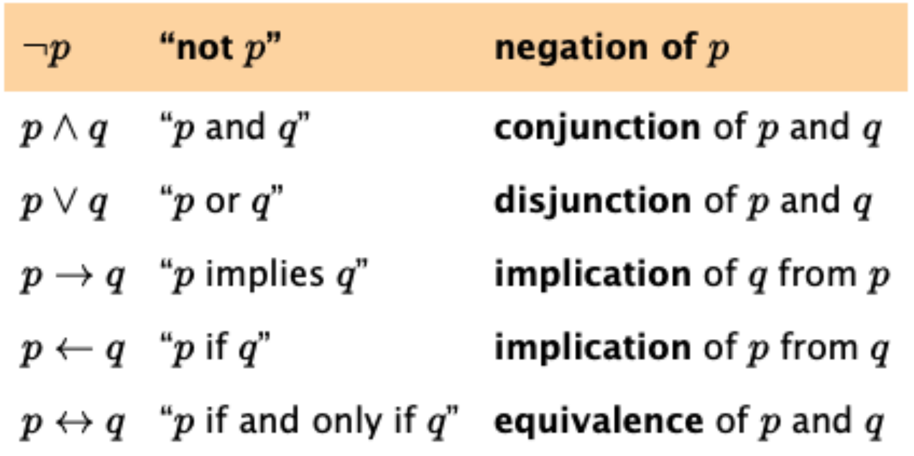
If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Propositional Logic: Syntax

A **proposition** is a sentence, written in a language, that has a truth value (i.e., it is true or false) in a world.

A proposition is built from atomic propositions using logical connectives.

- A proposition or logical formula is either
  - an **atomic proposition** or
  - a n



$\neg p$	"not p"	**negation** of p
$p \wedge q$	"p and q"	**conjunction** of p and q
$p \vee q$	"p or q"	**disjunction** of p and q
$p \rightarrow q$	"p implies q"	**implication** of q from p
$p \leftarrow q$	"p if q"	**implication** of p from q
$p \leftrightarrow q$	"p if and only if q"	**equivalence** of p and q

where p and q are propositions and operators $\neg$, $\wedge$, $\vee$, $\rightarrow$, $\leftarrow$ and $\leftrightarrow$ are **logical connectives**

Semantics

- An interpretation I assigns a truth value to each atom.
- A body b1 ∧ b2 is true in I if b1 is true in I and b2 is true in I.
- A rule h ⟵ b is false in I if b is true in I and h is false in I. The rule is true otherwise.
- A knowledge base KB is true in I if and only if every clause in KB is true in I.

Propositional logic: Semantics

Rules for evaluating truth with respect to a model *m*:

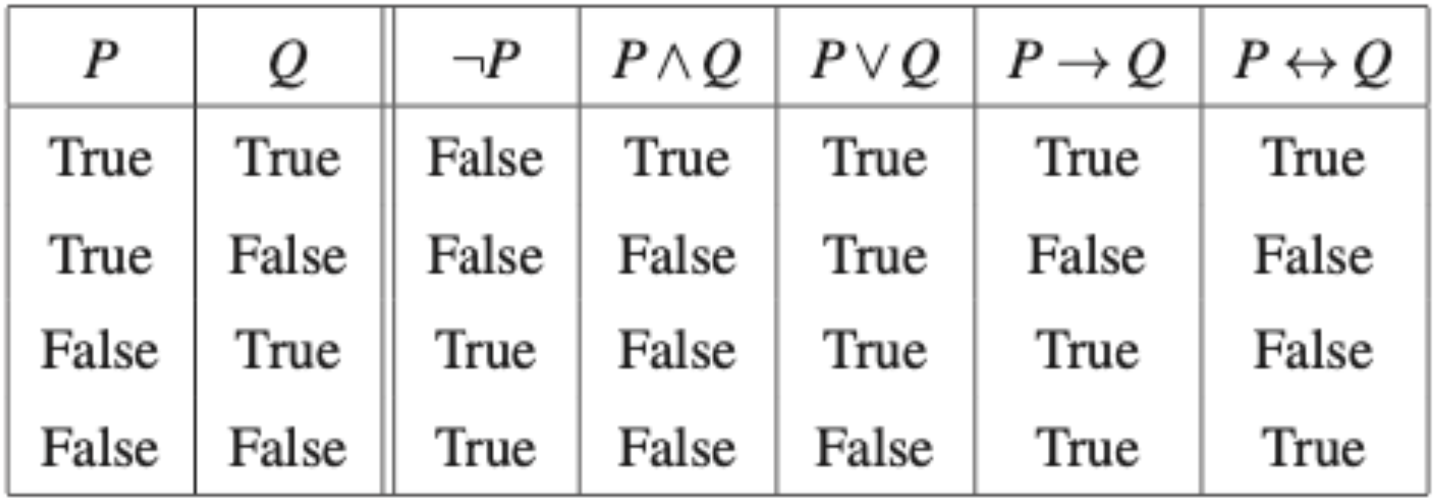
$\neg S$	is TRUE iff	S	is FALSE		
$S_1 \wedge S_2$	is TRUE iff	S_1	is TRUE and	S_2	is TRUE
$S_1 \vee S_2$	is TRUE iff	S_1	is TRUE or	S_2	is TRUE
$S_1 \Rightarrow S_2$	is TRUE iff	S_1	is FALSE or	S_2	is TRUE
i.e.	is FALSE iff	S_1	is TRUE and	S_2	is FALSE
$S_1 \Leftrightarrow S_2$	is TRUE iff	$S_1 \Rightarrow S_2$	is TRUE and	$S_2 \Rightarrow S_1$	is TRUE

Simple recursive process evaluates an arbitrary sentence, e.g.

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \text{TRUE} \wedge (\text{FALSE} \vee \text{TRUE}) = \text{TRUE} \wedge \text{TRUE} = \text{TRUE}$$

Semantics

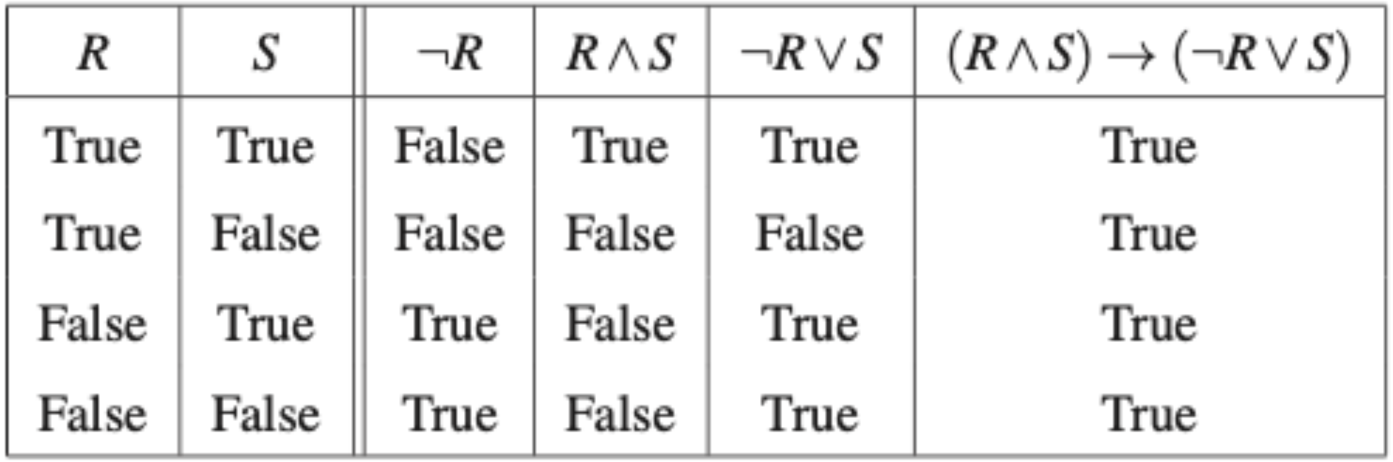
- The semantics of the connectives can be given by truth tables



P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
True	True	False	True	True	True	True
True	False	False	False	True	False	False
False	True	True	False	True	True	False
False	False	True	False	False	True	True

- One row for each possible assignment of True/False to variables
- **Important**: *P* and *Q* are any sentences, including complex sentences

Example – Complex Sentence



R	S	$\neg R$	$R \wedge S$	$\neg R \vee S$	$(R \wedge S) \rightarrow (\neg R \vee S)$
True	True	False	True	True	True
True	False	False	False	False	True
False	True	True	False	True	True
False	False	True	False	True	True

Thus $(R \wedge S) \rightarrow (\neg R \vee S)$ is a tautology

Entailment

Entailment means that one thing follows from another:

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true.

e.g. the KB containing "the Moon is full" and "the tide is high" entails "Either the Moon is full or the tide is high".

e.g. $x + y = 4$ entails $4 = x + y$

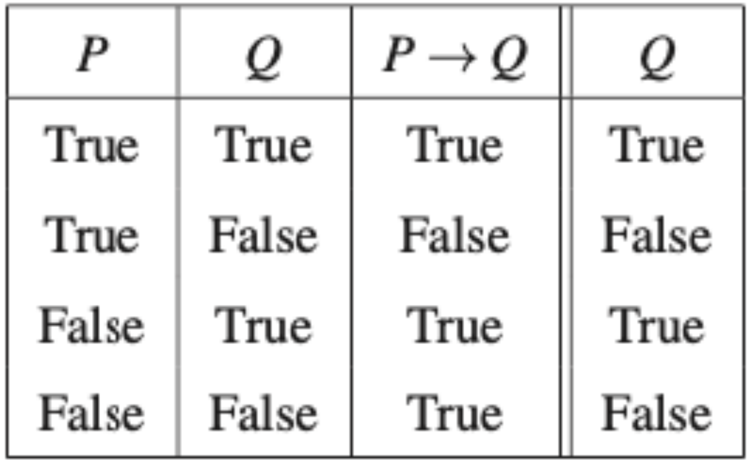
Entailment is a relationship between sentences (i.e. syntax) that is based on semantics.

Simple Entailments

Write $P \models Q$ for $\{P\} \models Q$

$P \wedge Q \models P$	$P \wedge Q \models Q$
$P \models P \vee Q$	$Q \models P \vee Q$
$P \models \neg\neg P$	$\neg\neg P \models P$
$\{P, P \rightarrow Q\} \models Q$	
If $P \models Q$ then $\models P \rightarrow Q$	

Entailment Example

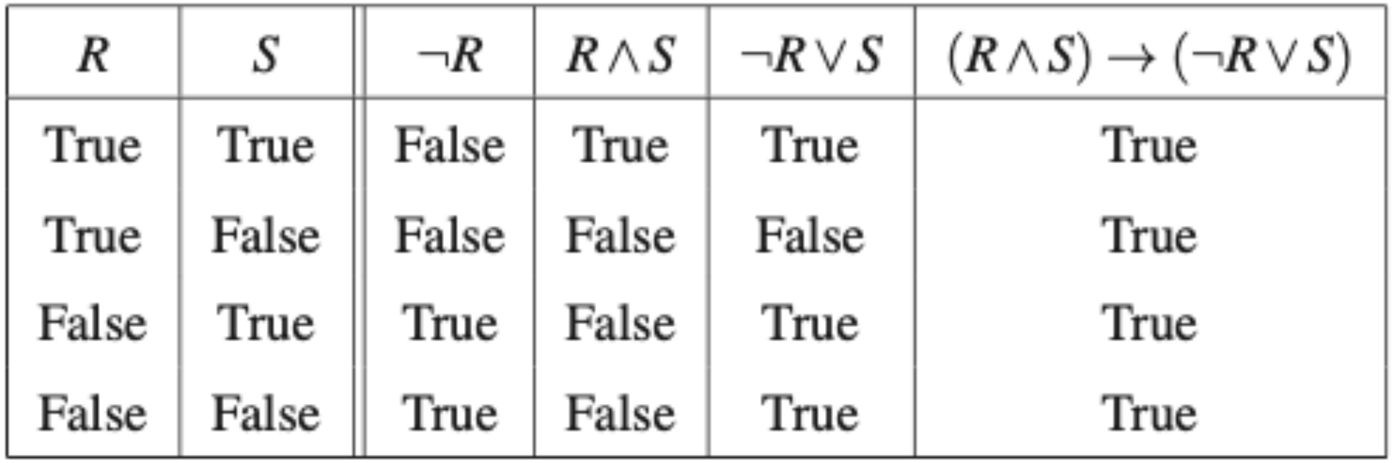


P	Q	$P \to Q$	Q
True	True	True	True
True	False	False	False
False	True	True	True
False	False	True	False

Therefore $\{P, P \to Q\} \models Q$

- In the only row where both *P* and *P*→*Q* are True(row1), *Q*is also True(here *S* is the set{*P*, *P*→*Q*})
- Note: The column for *P* → *Q* is calculated from that for *P* and *Q* using the truth table definition, and *Q* is used again to check the entailment

Entailment – Tautology



R	S	$\neg R$	$R \wedge S$	$\neg R \vee S$	$(R \wedge S) \rightarrow (\neg R \vee S)$
True	True	False	True	True	True
True	False	False	False	False	True
False	True	True	False	True	True
False	False	True	False	True	True

Therefore $\models (R \wedge S) \rightarrow (\neg R \vee S)$

Knowledge base

- A **knowledge base** is a set of propositions that are stated to be true.
  - An element of the knowledge base is an **axiom**.

Models

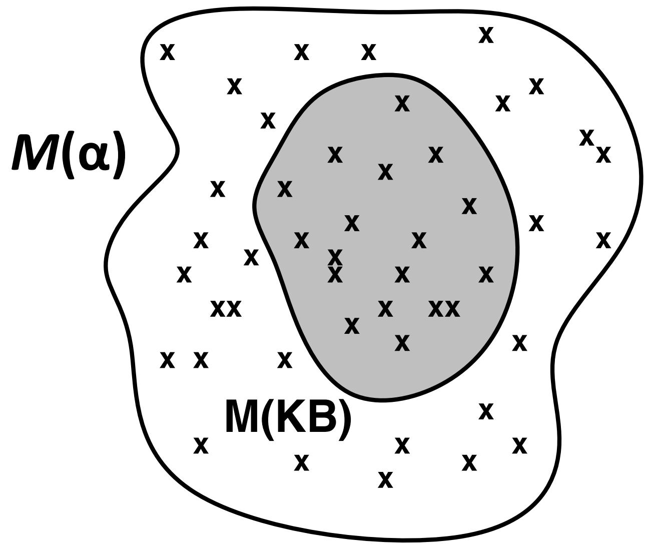
Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated.

- For Propositional Logic, a model is one row of the truth table

We say *m* is a model of a sentence α if α is true in *m*

$M(\alpha)$ is the set of all models of α

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$



Models and Logical Consequence

- A model of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written

  KB $\models$ g, if g is true in every model of KB.
- That is, KB $\models$ g if there is no interpretation in which KB is true and g is false.

Models and Logical Consequence

- A model of a set of clauses is an interpretation in which all the clauses are true.
- If KB is a set of clauses and g is a conjunction of atoms, g is a logical consequence of KB, written

  KB $\models$ g, if g is true in every model of KB.
- That is, KB $\models$ g if there is no interpretation in which KB is true and g is false.

$\models$

If KB $\models$ g we also say g **logically follows** from KB, or KB **entails** g.

Truth Tables - example

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$
F	F	T	F	F	T
F	T	T	F	T	T
T	F	F	F	T	F
T	T	F	T	T	T

P	"Fred is served alcohol"
Q	"Fred is over 18 years old"
$P \Rightarrow Q$	"If Fred is served alcohol, then he must be over 18"

Implication is not a causal relationship, but a rule that needs to be checked.

Logical Equivalence and Inference Rules

Inference Rules:

generalization:	$p \Rightarrow p \vee q$
specialization:	$p \wedge q \Rightarrow p$

Two sentences are logically equivalent iff true in same models: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

Logical Equivalence Rules

commutativity:	$p \wedge q \Leftrightarrow q \wedge p$	$p \vee q \Leftrightarrow q \vee p$
associativity:	$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$	$p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
distributivity:	$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
implication:	$(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$	
idempotent:	$p \wedge p \Leftrightarrow p$	$p \vee p \Leftrightarrow p$
double negation:	$\neg\neg p \Leftrightarrow p$	
contradiction:	$p \wedge \neg p \Leftrightarrow \text{FALSE}$	
excluded middle:		$p \vee \neg p \Leftrightarrow \text{TRUE}$
de Morgan:	$\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$	$\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$

Validity and Satisfiability

A sentence is valid if it is true in all models,
e.g. TRUE, $A \lor \neg A$, $A \Rightarrow A$, $(A \land (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:
$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in some model
e.g. $(A \lor B) \land C$

A sentence is unsatisfiable if it is true in no models
e.g. $A \land \neg A$

Satisfiability is connected to inference via the following:
$KB \models \alpha$ if and only if $(KB \land \neg\alpha)$ is unsatisfiable
i.e. prove α by *reductio ad absurdum*

Logical Equivalence Rules

commutativity:	$p \wedge q \Leftrightarrow q \wedge p$	$p \vee q \Leftrightarrow q \vee p$
associativity:	$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$	$p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$
distributivity:	$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$
implication:	$(p \Rightarrow q) \Leftrightarrow (\neg p \vee q)$	
idempotent:	$p \wedge p \Leftrightarrow p$	$p \vee p \Leftrightarrow p$
double negation:	$\neg\neg p \Leftrightarrow p$	
contradiction:	$p \wedge \neg p \Leftrightarrow \text{FALSE}$	
excluded middle:		$p \vee \neg p \Leftrightarrow \text{TRUE}$
de Morgan:	$\neg(p \wedge q) \Leftrightarrow (\neg p \vee \neg q)$	$\neg(p \vee q) \Leftrightarrow (\neg p \wedge \neg q)$

Example

Prove that:

$$(A \wedge (B \Rightarrow C)) \;\Leftrightarrow\; (\neg(A \Rightarrow B) \vee (A \wedge C))$$

$$\begin{aligned}
(A \wedge (B \Rightarrow C)) \quad &\Leftrightarrow \quad A \wedge (\neg B \vee C) && \text{[implication]}\\
&\Leftrightarrow \quad (A \wedge \neg B) \vee (A \wedge C) && \text{[distributivity]}\\
&\Leftrightarrow \quad \neg(\neg A \vee B) \vee (A \wedge C) && \text{[de Morgan]}\\
&\Leftrightarrow \quad \neg(A \Rightarrow B) \vee (A \wedge C) && \text{[implication]}
\end{aligned}$$

Proof methods

Proof methods divide into (roughly) two kinds:

- Model checking
  - truth table enumeration (always exponential in *n*)
  - heuristic search in model space (sound but incomplete)
  - improved backtracking, …..

- Application of inference rules
  - Legitimate (sound) generation of new sentences from old
  - Proof = a sequence of inference rule applications
    Can use inference rules as operators in a standard search algorithm
  - Typically require transformation of sentences into a normal form

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Soundness: *i* is sound if

whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: *i* is complete if

whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

.

Conjunctive Normal Form

In order to apply Resolution, we must first convert the KB into Conjunctive Normal Form (CNF).

This means that the KB is a conjunction of clauses, and each clause is a disjunction of (possibly negated) literals. e.g. $(A \vee \neg B) \wedge (B \vee \neg C \vee \neg D)$

Conversion to CNF

$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

1. Eliminate $\Leftrightarrow$, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.

$$(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$$

2. Eliminate $\Rightarrow$, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$.

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$$

3. Move $\neg$ inwards using de Morgan's rules and double-negation:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten:

$$(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$$

Resolution Rule: Key Idea

- Consider $A_1 \lor \cdots \lor A_m \lor B$ and $\neg B \lor C_1 \lor \cdots \lor Cn$
  - Suppose both are True
  - If B is True, $\neg B$ is False and $C1 \lor \cdots \lor Cn$ is True
  - If B is False, $A1 \lor \cdots \lor Am$ is True
  - Hence $A1 \lor \cdots \lor Am \lor C1 \lor \cdots \lor Cn$ is True
  - Hence the resolution rule is sound

- Starting with true premises, any conclusion made using resolution must be true

Resolution

- Another type of proof system based on refutation
- Better suited to computer implementation than systems of axioms and rules (can give correct ‘no’ answers)
- Decidable in the case of Propositional Logic
- Needs all formulae to be converted to clausal form

Resolution

$$\begin{tikzcd}[column sep=small]
A_1 \vee \cdots \vee A_m \vee B \arrow[dr, dash] & & \neg B \vee C_1 \vee \cdots \vee C_n \arrow[dl, dash] \\
& A_1 \vee \cdots \vee A_m \vee C_1 \vee \cdots \vee C_n &
\end{tikzcd}$$

where *B* is a propositional variable and *Ai* and *Cj* are literals

- *B* and ¬*B* are complementary literals
- $A1 \vee \cdots \vee Am \vee C1 \vee \cdots \vee Cn$ is the resolvent of the two clauses
- Special case: If no *Ai* and *C j* , resolvent is empty clause, $\square$

Resolution

Suppose we have two disjunctive clauses $\ell_1 \vee \ldots \vee \ell_i \vee \ldots \vee \ell_k$ and $m_1 \vee \ldots \vee m_j \vee \ldots \vee m_n$

We can then derive a new clause by eliminating l_i, m_j and combining all the other literals, i.e.

$$\frac{\ell_1 \vee \ldots \vee \ell_i \vee \ldots \vee \ell_k, \qquad m_1 \vee \ldots \vee m_j \vee \ldots \vee m_n}{\ell_1 \vee \ldots \vee \ell_{i-1} \vee \ell_{i+1} \vee \ldots \vee \ell_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where ℓ_i and m_j are complementary literals. e.g.

$$\frac{P_{1,3} \vee P_{2,2}, \qquad \neg P_{2,2}}{P_{1,3}}$$

Resolution is sound and complete for propositional logic.

Proof Methods

- Resolution provides us an alternative proof method which is generally somewhat faster than Truth Table Enumeration:

1. convert the into Conjunctive Normal Form,
2. add the negative of the clause you are trying to prove,
3. continually apply a series of resolutions until either
   - (a) you derive the empty clause, or
   - (b) no more pairs of clauses to which resolution can be applied

Horn Clauses

Model Checking can be done more efficiently if the clauses happen to be in a special form, for example, they may all be Horn Clauses.

Each Horn Clause is an implication involving only positive literals, in the form:

(conjunction of symbols) ⇒ symbol

e.g. $C \land (B \Rightarrow A) \land (C \land D \Rightarrow B)$

Deduction with Horn Clauses can be done by Modus Ponens:

$$\frac{p_1, \ldots, p_n, \qquad p_1 \land \ldots \land p_n \Rightarrow q}{q}$$

Efficient Proof Methods, using Horn Clauses, can generally be divided into Forward Chaining and Backward Chaining.

These algorithms are very natural and run in linear time!

Forward chaining

Look for a rule $p_1 \wedge \ldots \wedge p_n \Rightarrow q$ such that all the clauses on the left hand side are already in the KB.

Apply this rule, and add q to the KB.

Repeat this process until the goal clause α has been derived
(or we run out of rules to apply).

Proof of completeness

Forward chaining (FC) derives every atomic sentence that is entailed by *KB.*

1. FC reaches a fixed point where no new atomic sentences are derived
2. Consider the final state as a model *m*, assigning true/false to symbols
3. Every clause in the original *KB* is true in *m*

$$a_1 \wedge \ldots \wedge a_k \Rightarrow b$$

4. Hence *m* is a model of *KB*

1. If $KB \models q$, q is true in every model of *KB*, including *m*

Example

Forward chaining is a data driven method of deriving a particular goal from a given knowledge base and set of inference rules

- Inference rules are applied by matching facts to the antecedents of consequence relations in the knowledge base
- The application of inference rules results in new knowledge (from the consequents of the relations matched), which is then added to the knowledge base

I. Bratko , Prolog

Condition – action rules example

”Production rule” is a synonym for "rule", i.e. for a condition- action rule.
“If Then rules” are also called **condition-action rules**.

These components of a rule-based system have the form:
if <condition> **then** <conclusion>
or
if <condition**>** **then** <action>

Example:
if patient has high levels of the enzyme ferritin in their blood
and patient has the Cys282→Tyr mutation in HFE gene
then conclude patient has haemochromatosis*
* medical validity of this rule is not asserted here

Rules can be evaluated by:

- backward chaining or
- forward chaining

Example

Forward chaining is a data driven method of deriving a particular goal from a given knowledge base and set of inference rules

- Inference rules are applied by matching facts to the antecedents of consequence relations in the knowledge base

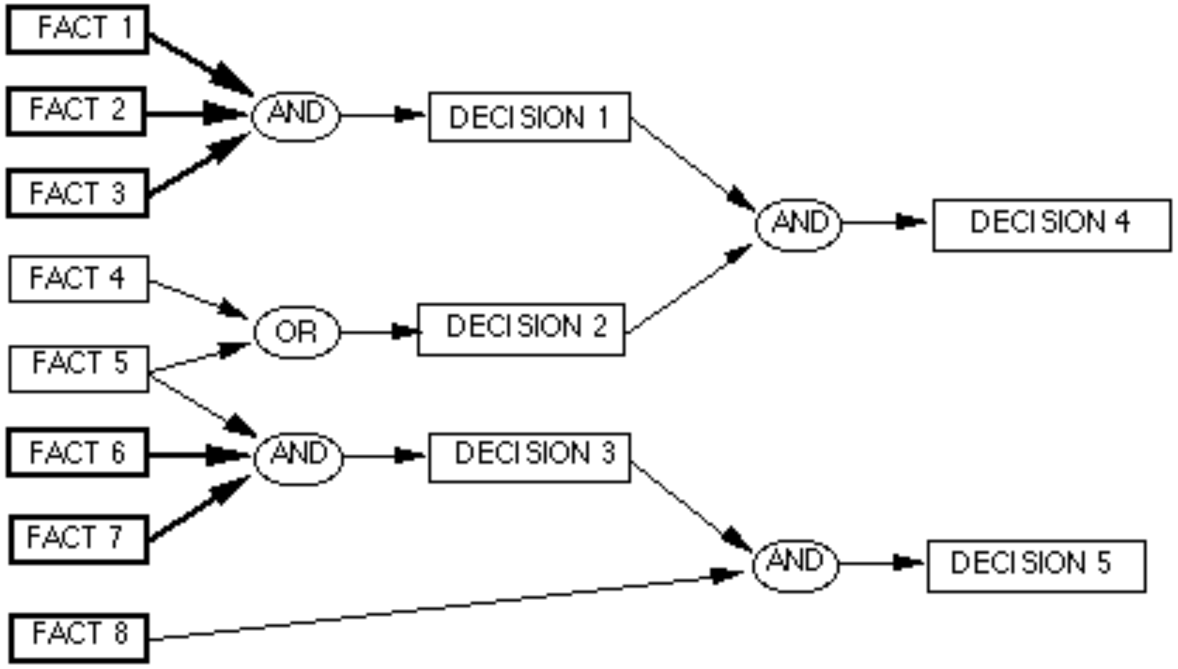
- The application of inference rules results in new knowledge (from the consequents of the relations matched), which is then added to the knowledge base

I. Bratko , Prolog

Forward Chaining

Given some facts, work forward through inference network.

Discovers what conclusions can be derived from data.



Backward chaining

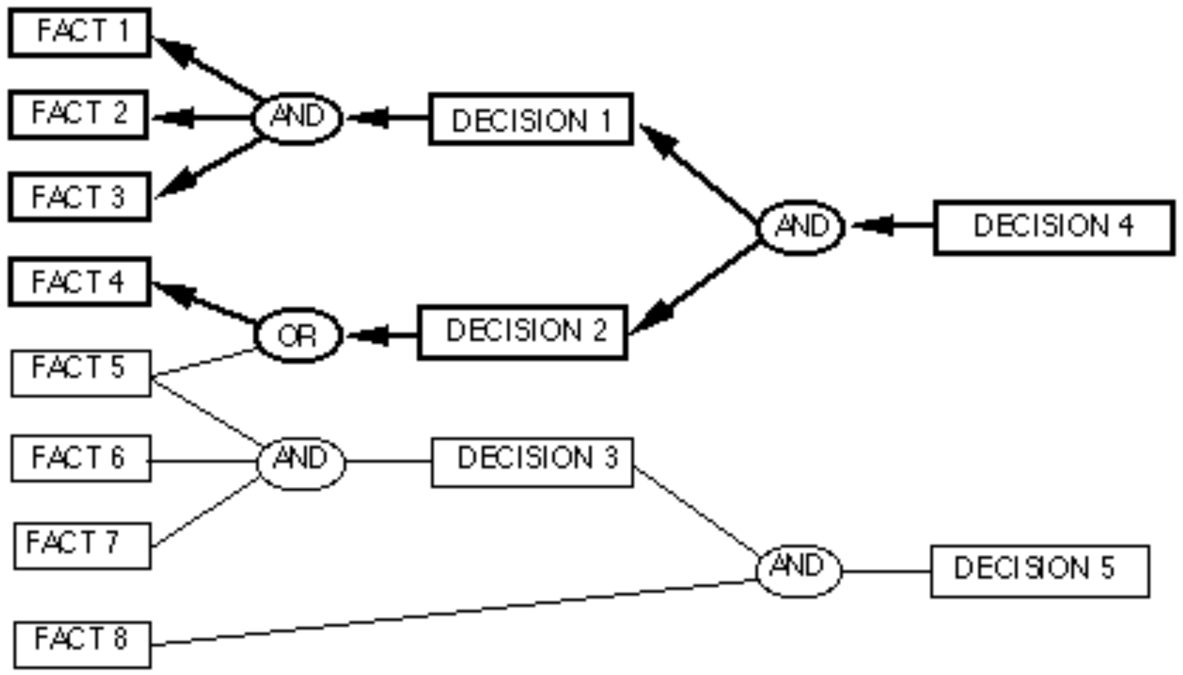
Backward Chaining instead maintains a list of subgoals that it is trying to prove. Initially, this list consists of the ultimate goal α.

Choose a clause *q* from the list of subgoals.

- check if *q* is known already
- otherwise, find a rule with *q* on the right side and add clauses from the left side of this rule as new subgoals
- check to make sure each new subgoal is not on the list already, and has not already been proved, or failed

Backward Chaining

- To determine if a decision should be made, work backwards looking for justifications for the decision. Eventually, a decision must be justified by facts.



Forward vs. Backward Chaining

Forward Chaining is data-driven automatic, unconscious processing e.g. object recognition, routine decisions

- May do lots of work that is irrelevant to the goal

Backward Chaining is goal-driven, appropriate for problem-solving

- e.g. Where are my keys? How do I get into a PhD program?

Complexity of Backward Chaining can be much less than linear in size of KB

Satisfiability as Constraint Satisfaction

Suppose you are given a KB written in 3-CNF. (This means Conjunctive Normal Form, with at most three literals in each clause.)

Does there exist any assignment of truth values to the symbols which will make all of the clauses in the KB TRUE?

For example, is there an assignment of truth values to *A*,*B*,*C*,*D*,*E* which will make the following TRUE?

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

This provides a classic example of a Constraint Satisfaction Problem

Propositional Logic

- Propositions built from $\wedge, \vee, \neg, \rightarrow$
- Sound, complete and decidable proof systems (inference procedures)
  - Natural deduction
  - Resolution refutation
  - Prolog for special case of definite clauses
  - Tableau method
- Limited expressive power
  - Cannot express ontologies
- First-Order Logic can express knowledge about objects, properties and relationships between objects

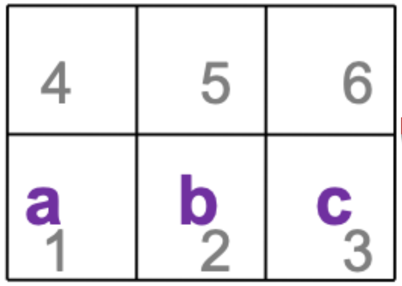
Logics in General

Language	Ontology	Epistemology
Propositional logic	facts	true / false / unknown
First-order logic	facts, objects, relations	true / false / unknown
Temporal logic	facts, objects, relations, times	true / false / unknown
Probability theory	facts	degree of belief
Fuzzy logic	facts + degree of truth	known interval value

Syntax of First Order Logic

- Objects: people, houses, numbers, theories, colors, football games, wars, centuries ...
- Predicates: red, round, bogus, prime, multistoried, ... brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, ...
- Functions: father of, best friend, third inning of, one more than,...

Syntax of First Order Logic



Cels

Robots: a, b, c, cells 1, ..., 6

In the robors world:

Constants	a, b, c, c1, c2, [1,2],[2,1], etc.
Predicates	at(a,1), at(b,2) … at(R,A)
Functions	Resulti()
Variables	x, y, a, t ….
Connectives	$\wedge \vee \neg \Rightarrow \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall \exists$

Syntax is same as in propositional logics plus the Quantifies

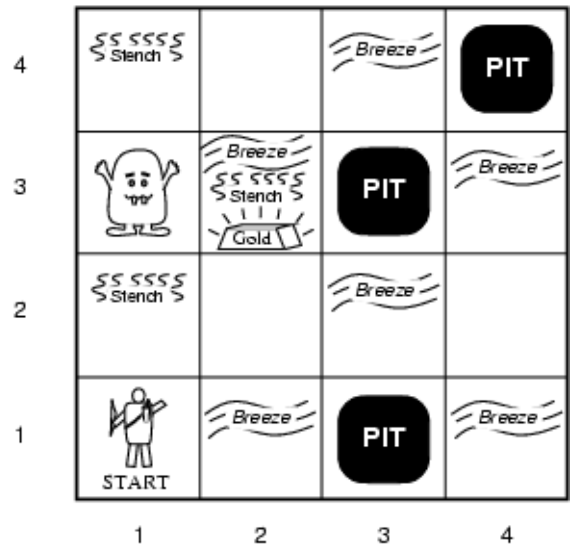
Syntax of First Order Logic

In the Wampus world:

Squares

Constants	$Gold, Wumpus, [1,2], [3,1]$, etc.
Predicates	$Adjacent(), Smell(), Breeze(), At()$
Functions	$Result()$
Variables	$x,\ y,\ a,\ t, \ldots$
Connectives	$\wedge\ \vee\ \neg\ \Rightarrow\ \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Quantifiers



Wumpus World

Syntax is same as in propositional logics plus the Quantifies

Sentences

Atomic sentence $= \mathit{predicate}(\mathit{term}_1, \ldots, \mathit{term}_n)$
or $\mathit{term}_1 = \mathit{term}_2$

Term $= \mathit{function}(\mathit{term}_1, \ldots, \mathit{term}_n)$
or *constant* or *variable*

e.g. $\mathit{At}(\mathit{Agent}, [1,1], S_0)$ (S$_o$ – situation at the start)

$\mathit{Holding}(\mathit{Gold}, S_5)$

Complex sentences are made from atomic sentences using connectives

$$\neg S, \quad S_1 \wedge S_2, \quad S_1 \vee S_2, \quad S_1 \Rightarrow S_2, \quad S_1 \Leftrightarrow S_2$$

e.g. $\mathit{Pit}(x) \wedge \mathit{Adjacent}(x, y)$

Universal Quantification

$\forall\langle\text{variables}\rangle\langle\text{sentence}\rangle$

Where there's glitter, there's gold:

$\forall x\ Glitter(x) \Rightarrow At(Gold, x)$

$\forall x\ P$ is equivalent to the conjunction of instantiations of P

$$\begin{array}{ll} & Glitter([1,1]) \Rightarrow At(Gold,[1,1]) \\ \wedge & Glitter([1,2]) \Rightarrow At(Gold,[1,2]) \\ \wedge & Glitter([1,3]) \Rightarrow At(Gold,[1,3]) \\ \wedge & \ldots \end{array}$$

Universal Quantification

Typically, $\Rightarrow$ is the main connective with $\forall$

Common mistake: using $\wedge$ as the main connective with $\forall$

$$\forall x\, Glitter(x) \wedge At(Gold, x)$$

means "There is Glitter everywhere and Gold everywhere."

Existential Quantification

$\exists\langle\text{variables}\rangle\langle\text{sentence}\rangle$

Some sheep are black

$\exists x\; Sheep(x) \wedge Black(x)$

$\exists x\; P$ is equivalent to the disjunction of instantiations of P

$$\begin{aligned} & Sheep(Dolly) \wedge Black(Dolly) \\ \vee\quad & Sheep(Lassie) \wedge Black(Lassie) \\ \vee\quad & Sheep(Skippy) \wedge Black(Skippy) \\ \vee\quad & \ldots \end{aligned}$$

Existential Quantification

Typically, $\wedge$ is the main connective with $\exists$

Common mistake: using $\Rightarrow$ as the main connective with $\exists$

$$\exists x\, Sheep(x) \Rightarrow Black(x)$$

is true if there is anyone who is not at sheep!

Nested Quantifiers

The order of quantification is very important

- Everything likes everything — $\forall x \forall y\, likes(x, y)$ (or $\forall y \forall x\, likes(x, y)$)
- Something likes something — $\exists x \exists y\, likes(x, y)$ (or $\exists y \exists x\, likes(x, y)$)
- Everything likes something — $\forall x \exists y\, likes(x, y)$
- There is something liked by everything — $\exists y \forall x\, likes(x, y)$

Properties of Quantifiers

$\forall x \forall y$ is the same as $\forall y \forall x$ (Why?)

$\exists x \exists y$ is the same as $\exists y \exists x$ (Why?)

$\exists x \forall y$ is not the same as $\forall y \exists x$

$\exists x \forall y\, Loves(x, y)$
"There is a person who loves everyone in the world"

$\forall y \exists x\, Loves(x, y)$
"Everyone in the world is loved by at least one person"

Quantifier duality: each can be expressed using the other

$\forall x\, Likes(x, IceCream) \qquad \neg \exists x\, \neg Likes(x, IceCream)$

$\exists x\, Likes(x, Broccoli) \qquad \neg \forall x\, \neg Likes(x, Broccoli)$

Fun with Sentences

Brothers are siblings

“Sibling” is symmetric

One’s mother is one’s female parent

A first cousin is a child of a parent’s sibling

Fun with Sentences

Brothers are siblings

$$\forall x,y\ Brother(x,y) \Rightarrow Sibling(x,y)$$

“Sibling” is symmetric

$$\forall x,y\ Sibling(x,y) \Leftrightarrow Sibling(y,x)$$

One’s mother is one’s female parent

$$\forall x,y\ Mother(x,y) \Leftrightarrow (Female(x) \wedge Parent(x,y))$$

A first cousin is a child of a parent’s sibling

$$\forall x,y FirstCousin(x,y) \Leftrightarrow \exists p,ps\ Parent(p,x) \wedge Sibling(p,q) \wedge Parent(q,y)$$

Deducing Hidden Properties

Properties of locations:

$\forall x,t\ At(Agent,x,t) \wedge Smell(t) \Rightarrow Smelly(x)$

$\forall x,t\ At(Agent,x,t) \wedge Breeze(t) \Rightarrow Breezy(x)$

$\forall x,t\ At(Agent,x,t) \wedge Glitter(t) \Rightarrow AtGold(x)$

Squares are breezy near a pit:

Causal rule – infer effect from cause

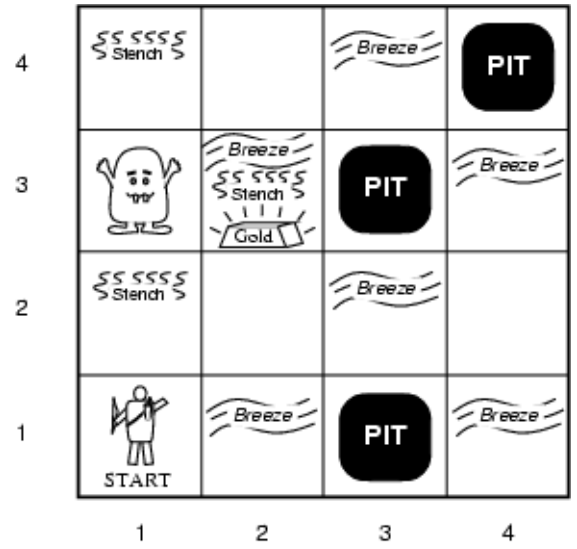
$\forall x,y\ Pit(x) \wedge Adjacent(x,y) \Rightarrow Breezy(y)$

Diagnostic rule – infer cause from effect

$\forall y\ Breezy(y) \Rightarrow \exists x\ Pit(x) \wedge Adjacent(x,y)$

Definition for the *Breezy* predicate (combines Causal and Diagnostic):

$\forall y\ Breezy(y) \Leftrightarrow [\exists x\ Pit(x) \wedge Adjacent(x,y)]$



Wumpus World

Describing Actions

- We can plan a series of actions in a logical domain in a manner analogous to the Path Search algorithms
- But, instead of the successor state being explicitly specified, we instead need to deduce what will be true and false in the state resulting from the previous state and action:
- Effect axiom describe changes due to action
  - Delivery robot example

Delivery robot - Plan Graph

Actions

mc: move clockwise

mac: move anticlockwise

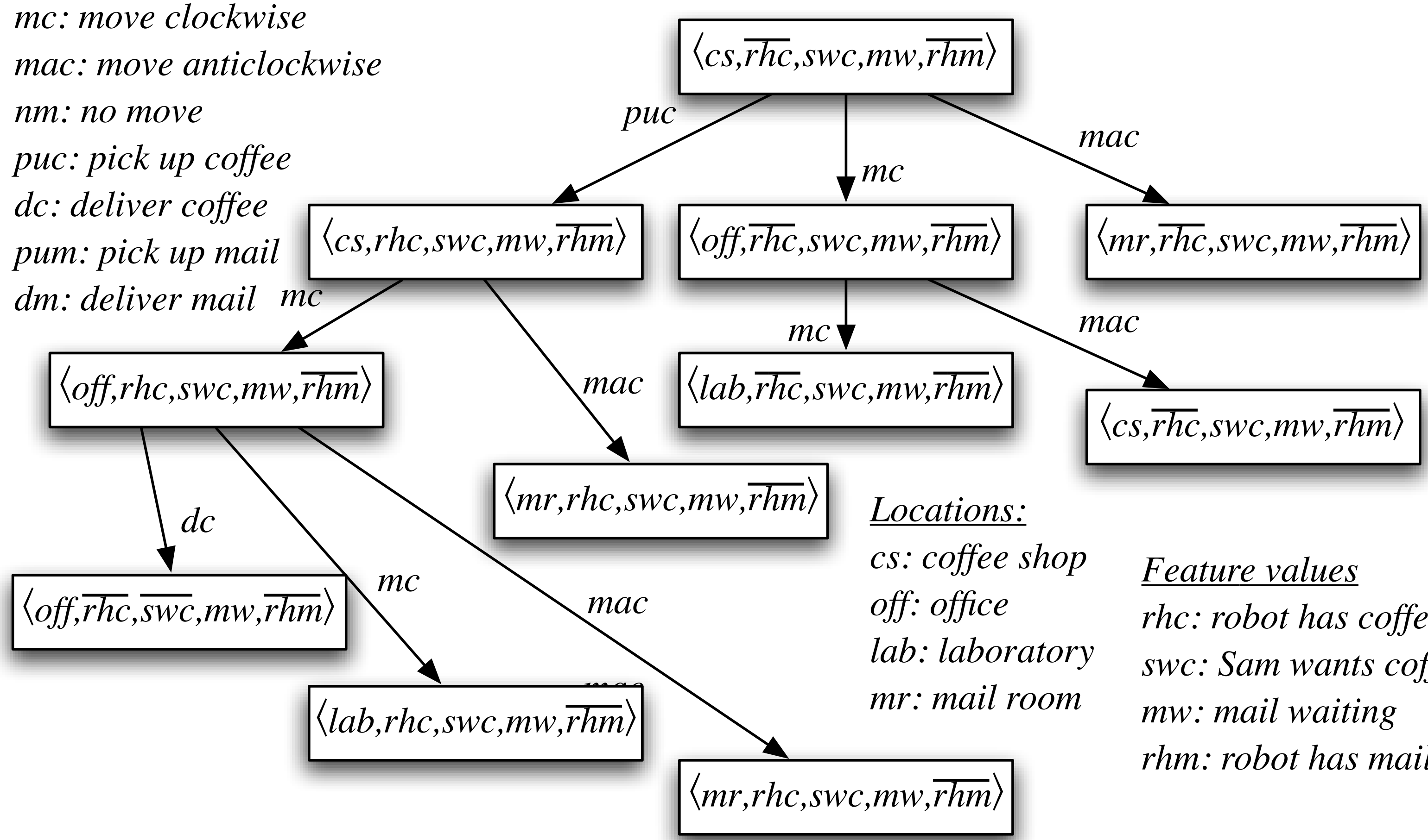
nm: no move

puc: pick up coffee

dc: deliver coffee

pum: pick up mail

dm: deliver mail


⟨cs,$\overline{rhc}$,swc,mw,$\overline{rhm}$⟩
puc
mc
mac
⟨cs,rhc,swc,mw,$\overline{rhm}$⟩
⟨off,$\overline{rhc}$,swc,mw,$\overline{rhm}$⟩
⟨mr,$\overline{rhc}$,swc,mw,$\overline{rhm}$⟩
mc
mac
mc
mac
⟨off,rhc,swc,mw,$\overline{rhm}$⟩
⟨mr,rhc,swc,mw,$\overline{rhm}$⟩
⟨lab,$\overline{rhc}$,swc,mw,$\overline{rhm}$⟩
⟨cs,$\overline{rhc}$,swc,mw,$\overline{rhm}$⟩
dc
mc
mac
⟨off,$\overline{rhc}$,$\overline{swc}$,mw,$\overline{rhm}$⟩
⟨lab,rhc,swc,mw,$\overline{rhm}$⟩
⟨mr,rhc,swc,mw,$\overline{rhm}$⟩

Locations:

cs: coffee shop

off: office

lab: laboratory

mr: mail room

Feature values

rhc: robot has coffee

swc: Sam wants coffee

mw: mail waiting

rhm: robot has mail

Describing Actions

Frame problem: Some facts will change as a result of an action, but many more will stay as they were.

However, adding too many of these frame axioms can make the process unmanageable.

- For example, if a cup is red, and you turn it upside down, it is still red.
- But, if a cup is full of water, and you turn it upside down, it is no longer full of water.

Large-scale expert systems of the 1980's often failed because of their inability to encode this kind of "commonsense" reasoning in explicit rules.

Describing Actions

- Qualification problem: Normally, we expect actions to have a certain effect. But, in the real world there could be endless caveats. What happens if the gold is slippery, or nailed down, or too heavy, or you can't reach it, etc.
- Ramification problem: Real actions have many secondary consequences – what about the dust on the gold, wear and tear on gloves, shoes, etc..
- In general, we assume that a fact is true if a rule tells us that an action made it true, or if it was true before and no action made it false.

Searching for a Situation

Initial condition in KB (knowledge base)

Query: *Ask - the robot has the coffee?*

i.e., in what situation will I be holding the coffee?

Answer :

i.e., go *acc* and then pickup the coffee

This assumes that the agent is interested in plans starting at S_0 and that S_0 is the only situation described in the KB.

Searching for a Plan of Actions

- Represent plans as action sequences $[a_1, a_2, \ldots, a_n]$
- Planning systems are special-purpose reasoners designed to do this type of inference more efficiently than a general-purpose reasoner.

Summary

Logical agents apply inference to a knowledge base to derive new information and make decisions.

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

Summary

- First-order logic is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the *KB*.

- Resolution is complete for propositional logic
  Forward, backward chaining are linear-time, complete for Horn clauses

- Propositional logic lacks expressive power

Summary

- Ambiguity of natural languages avoided with formal languages
- Enables formalization of (truth preserving) entailment
- Propositional Logic: Simplest logic of truth and falsity
- Knowledge Based Systems: First-Order Logic

Summary

- First Order Logic:
  objects and relations are semantic primitives
  syntax: constants, functions, predicates, equality, quantifiers

- Increased expressive power

- Situation calculus:
  - conventions for describing actions and change
  - can formulate planning as inference on a knowledge base

Summary

- Knowledge is contained in agents in the form of **sentences** in a **knowledge representation language** that are stored in a **knowledge base**.
- A knowledge-based agent is composed of a knowledge base and an inference mechanism. It operates by storing sentences about the world in its knowledge base, using the inference mechanism to infer new sentences, and using these sentences to decide what action to take.
- A representation language is defined by its **syntax**, which specifies the structure of sentences, and its **semantics**, which defines the **truth** of each sentence in each **possible world** or **model**.

Summary

- The relationship of **entailment** between sentences is crucial to our understanding of reasoning. A sentence α entails another sentence β if β is true in all worlds where α is true. Equivalent definitions include the **validity** of the sentence α ⇒ β and the **unsatisfiability** of the sentence α ∧ ¬β.
- Inference is the process of deriving new sentences from old ones. **Sound** inference algo- rithms derive *only* sentences that are entailed; **complete** algorithms derive *all* sentences that are entailed.
- **Propositional logic** is a simple language consisting of **proposition symbols** and **logical connectives**. It can handle propositions that are known true, known false, or completely unknown.

Summary

- The set of possible models, given a fixed propositional vocabulary, is finite, so entailment can be checked by enumerating models. Efficient **model-checking** inference algorithms for propositional logic include backtracking and local search methods and can often solve large problems quickly.
- **Inference rules** are patterns of sound inference that can be used to find proofs. The **resolution** rule yields a complete inference algorithm for knowledge bases that are expressed in **conjunctive normal form**. **Forward chaining** and **backward chaining** are very natural reasoning algorithms for knowledge bases in **Horn form**.

References

- Poole &Mackworth, Artificial Intelligence: Foundations of Computational Agents, Chapter 5.
- Russell & Norvig, *Artificial Intelligence: a Modern Approach*, Chapter 6 &7.
- Ivan Bratko, *Prolog Programming for Artificial Intelligence*.

Additional material

Conversion to Conjunctive Normal Form

Eliminate $\leftrightarrow$ rewriting $P \leftrightarrow Q$ as $(P \rightarrow Q) \wedge (Q \rightarrow P)$

Eliminate $\rightarrow$ rewriting $P \rightarrow Q$ as $\neg P \vee Q$

Use De Morgan's laws to push $\neg$ inwards (repeatedly)

- Rewrite $\neg(P \wedge Q)$ as $\neg P \vee \neg Q$
- Rewrite $\neg(P \vee Q)$ as $\neg P \wedge \neg Q$

Eliminate double negations: rewrite $\neg\neg P$ as P

Use the distributive laws to get CNF [or DNF] – if necessary

- Rewrite $(P \wedge Q) \vee R$ as $(P \vee R) \wedge (Q \vee R)$ [for CNF]
- Rewrite $(P \vee Q) \wedge R$ as $(P \wedge R) \vee (Q \wedge R)$ [for DNF]

Applying Resolution: Naive Method

- Convert knowledge base into clausal form
- Repeatedly apply resolution rule to the resulting clauses
- *P* follows from the knowledge base if and only if each clause in the CNF of *P* can be derived using resolution from the clauses of the knowledge base (or subsumption)
- Example $\{P \to Q, Q \to R\} \vdash P \to R$

  Clauses $\neg P \vee Q$, $\neg Q \vee R$, show $\neg P \vee R$

  Follows from one resolution step

Applying Resolution Refutation

- Negate query to be proven (resolution is a refutation system)
- Convert knowledge base and negated query into CNF
- Repeatedly apply resolution until either the empty clause (contradiction) is derived or no more clauses can be derived
- If the empty clause is derived, answer ‘yes’ (query follows from knowledge base), otherwise answer ‘no’ (query does not follow from knowledge base)

Definitions

A sentence is valid if it is True under all possible assignments of True/False to its variables (e.g. $P \vee \neg P$)

A tautology is a valid sentence

Two sentences are equivalent if they have the same truth table, e.g. $P \wedge Q$ and $Q \wedge P$

- So P is equivalent to Q if and only if $P \leftrightarrow Q$ is valid

A sentence is satisfiable if there is some assignment of True/False to its variables for which the sentence is True

A sentence is unsatisfiable if it is not satisfiable (e.g. $P \wedge \neg P$)

- Sentence is False for all assignments of True/False to its variables
- So P is a tautology if and only if $\neg P$ is unsatisfiable

Material Implication

$P \to Q$ evaluates to False only when P is True and Q is False

$P \to Q$ is equivalent to $\neg P \vee Q$: material implication

English usage often suggests a causal connection between antecedent (P) and consequent (Q) – this is not reflected in the truth table

Examples

- $(P \wedge Q) \to Q$ is a tautology for any Q
- $P \to (P \vee Q)$ is a tautology for any Q
- $(P \wedge \neg P) \to Q$ is a tautology for any Q

Proof of Equivalence

Let $P \Leftrightarrow Q$ mean "P is equivalent to Q" ($P \Leftrightarrow Q$ is **not** a formula)

Then $P \wedge (Q \rightarrow R) \ \Leftrightarrow \ \neg(P \rightarrow Q) \vee (P \wedge R)$

$$\begin{array}{rcll}
P \wedge (Q \rightarrow R) & \Leftrightarrow & P \wedge (\neg Q \vee R) & \text{[Implication]} \\
 & \Leftrightarrow & (P \wedge \neg Q) \vee (P \wedge R) & \text{[Distributivity]} \\
 & \Leftrightarrow & (\neg\neg P \wedge \neg Q) \vee (P \wedge R) & \text{[Double negation]} \\
 & \Leftrightarrow & \neg(\neg P \vee Q) \vee (P \wedge R) & \text{[De Morgan]} \\
 & \Leftrightarrow & \neg(P \rightarrow Q) \vee (P \wedge R) & \text{[Implication]}
\end{array}$$

Assumes substitution: if $A \Leftrightarrow B$, replace A by B in any subformula

Assumes equivalence is transitive: if $A \Leftrightarrow B$ and $B \Leftrightarrow C$ then $A \Leftrightarrow C$

Entailment

S entails P ($S \models P$) if whenever all formulae in S are True, P is True

- Semantic definition – concerns truth (not proof)

Compute whether $S \models P$ by calculating a truth table for S and P

- Syntactic notion – concerns computation/proof
- Not always this easy to compute (how inefficient is this?)

A tautology is a special case of entailment where S is the empty set

- All rows of the truth table are True

Entailment Example

P	Q	$P \to Q$	Q
True	True	True	True
True	False	False	False
False	True	True	True
False	False	True	False

Therefore $\{P, P \to Q\} \models Q$

- In the only row where both P and $P \to Q$ are True (row 1), Q is also True (here S is the set $\{P, P \to Q\}$)

Note: The column for $P \to Q$ is calculated from that for P and Q using the truth table definition, and Q is used again to check the entailment