Applied AI

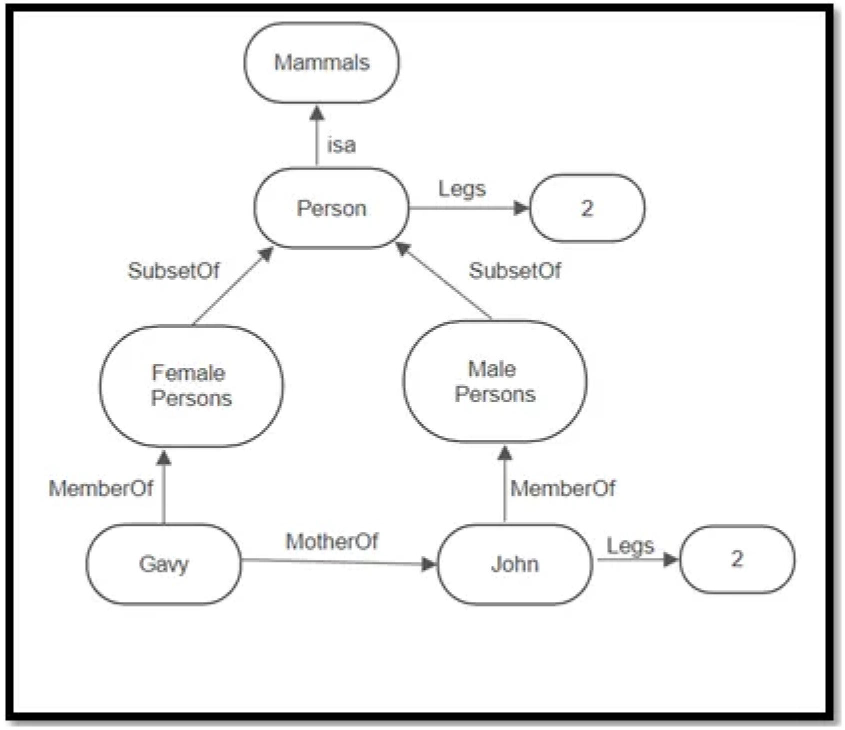
Lecture 5

Dr Artie Basukoski

[slides adapted from Artificial Intelligence: A Modern Approach, Russel and Norvigl]

Agenda

- Knowledge representation and reasoning
- Propositional logic
- Forward and Backward chaining
- Resolution
- First-order logic
- Logic programming
- Prolog systems



Knowledge representation

1. **Logical Representation:** Using formal logic to ensure precision and support inference.
   - **Propositional Logic**: Basic true/false statements.
   - **First-Order Logic**: Extends propositional logic with objects, relations, and quantifiers for more expressive power.
2. **Rule-Based Representation:**Uses condition-action rules to encode knowledge for decision-making.
   - **Production Rules**: "If-then" rules that trigger actions based on conditions.
   - **Constraint Satisfaction Problems (CSPs)**: Variables and constraints that must be satisfied for valid solutions.
3. **Graphical Representation:** Uses graphs or networks to represent concepts and their relationships visually
   - **Semantic Networks**: Concepts are nodes, and relationships are edges between them.
   - **Bayesian Networks**: Probabilistic relationships modeled with directed graphs for reasoning under uncertainty.
   - **Markov Networks**: Undirected graphs that capture local dependencies among variables.
4. **Structured Representation**: Represents knowledge using structured data formats.
   - **Frames**: Data structures with slots and values that represent stereotypical situations.
   - **Ontologies**: Formal definitions of concepts and relationships within a domain, often used with description logics.

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

E.g., 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

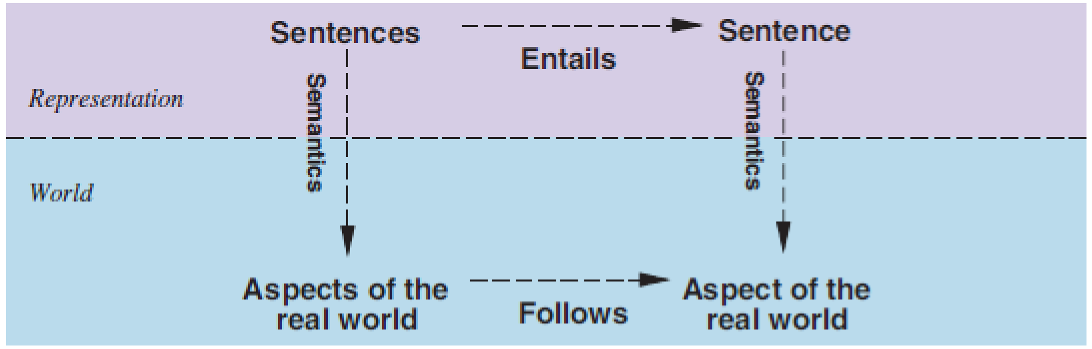
$x + 2 \geq y$ is true in a world where $x = 7,\ y = 1$
$x + 2 \geq y$ is false in a world where $x = 0,\ y = 6$

Entailment

Entailment means that one thing follows from another:

$KB \models a$

Knowledge base KB entails sentence a
if and only if
a is true in all worlds where KB is true



E.g., if the KB contains "It is raining" and "the grass is wet" then the KB entails "Either it is raining or the grass is wet"

E.g., $x + y = 4$ entails $4 = x + y$

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

Note: brains process syntax (of some sort)

Entailment is crucial for **reasoning** and **proving** things in logic. It allows us to derive new truths from established truths, which is a core aspect of logical reasoning.

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: Semantics

A model is a specific interpretation that makes the sentence true. Each **model** specifies true/false for each proposition symbol.

E.g.	P_1	P_2	P_3
	$true$	$true$	$false$

(With these symbols, 8 possible models, can be enumerated automatically.)

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.,

$P_1 \wedge (\neg P_2 \vee \neg P_3)$ = $true \wedge (false \vee true) = true \wedge true = true$

Truth tables for connectives

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*

This is known as inference by Enumeration (complete but exponential - slow)
E.g. KB = P ^ Q
Does KB ⊨P?
Yes. P is entailed by KB if all models of KB are models of P. In all rows where KB is true, P is also true.
Does KB ⊨¬P?
No. ¬P is not true in all rows where KB is true.

Logical equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$(\alpha \land \beta)$	$\equiv$	$(\beta \land \alpha)$	commutativity of $\land$
$(\alpha \lor \beta)$	$\equiv$	$(\beta \lor \alpha)$	commutativity of $\lor$
$((\alpha \land \beta) \land \gamma)$	$\equiv$	$(\alpha \land (\beta \land \gamma))$	associativity of $\land$
$((\alpha \lor \beta) \lor \gamma)$	$\equiv$	$(\alpha \lor (\beta \lor \gamma))$	associativity of $\lor$
$\neg(\neg\alpha)$	$\equiv$	α	double-negation elimination
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\beta \Rightarrow \neg\alpha)$	contraposition
$(\alpha \Rightarrow \beta)$	$\equiv$	$(\neg\alpha \lor \beta)$	implication elimination
$(\alpha \Leftrightarrow \beta)$	$\equiv$	$((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha))$	biconditional elimination
$\neg(\alpha \land \beta)$	$\equiv$	$(\neg\alpha \lor \neg\beta)$	De Morgan
$\neg(\alpha \lor \beta)$	$\equiv$	$(\neg\alpha \land \neg\beta)$	De Morgan
$(\alpha \land (\beta \lor \gamma))$	$\equiv$	$((\alpha \land \beta) \lor (\alpha \land \gamma))$	distributivity of $\land$ over $\lor$
$(\alpha \lor (\beta \land \gamma))$	$\equiv$	$((\alpha \lor \beta) \land (\alpha \lor \gamma))$	distributivity of $\lor$ over $\land$

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:

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

A sentence is satisfiable if it is true in some model

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

A sentence is unsatisfiable if it is true in no models

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

Inference

- Satisfiability is connected to inference via the following:
- $KB \models a$ if and only if $(KB \wedge \neg a)$ is unsatisfiable i.e.,

prove a by *reductio ad absurdum*

- KB: All birds have feathers; Tweety is a bird.
- α: Tweety has feathers.

1. Assume the opposite: Tweety does NOT have feathers (¬α).
2. Combine: (All birds have feathers) AND (Tweety is a bird) AND (Tweety does NOT have feathers). This is unsatisfiable! It contradicts itself.
3. Conclusion: Since assuming the opposite led to a contradiction, our original statement (α: Tweety has feathers) must be true.

Proof methods

Proof methods divide into (roughly) two kinds:

Application of inference rules

– Legitimate (sound) generation of new sentences from old

– Proof = a sequence of inference rule applications

  Can use inference rules in a standard search algorithm

– Typically require translation of sentences into a normal form, eg Horn clauses for backward chaining, CNF for resolution

Model checking

truth table enumeration (always exponential in n)

Explores all possible paths

heuristic search in model space (sound but incomplete)

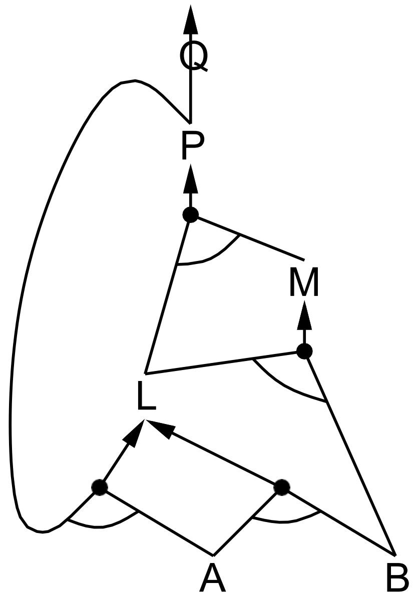
  e.g., hill-climbing algorithms

Forward chaining

Idea: fire any rule whose premises are satisfied in the KB,
add its conclusion to the KB, until query is found

E.g. Does this KB entail Q. Can Q be proven.

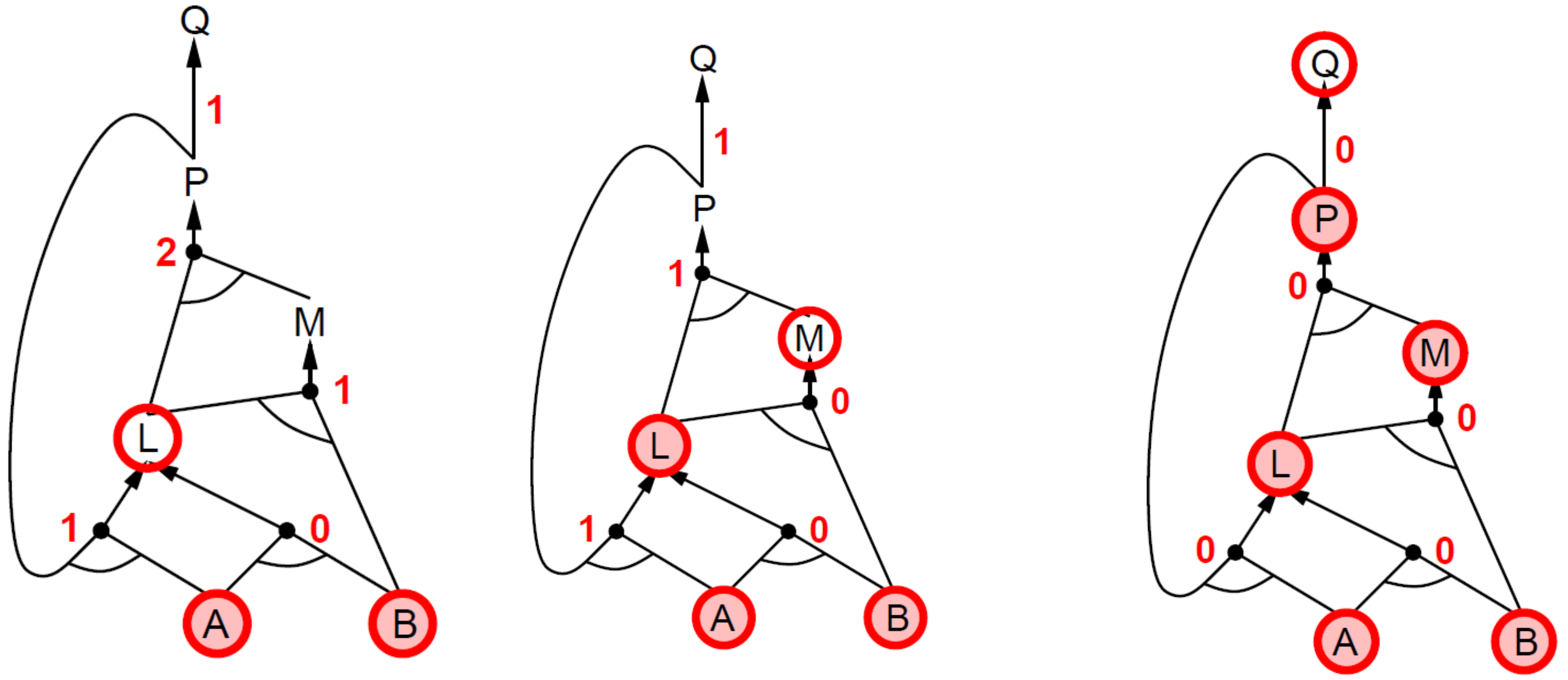
$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$



Forward chaining

Idea: fire any rule whose premises are satisfied in the KB,
add its conclusion to the KB, until query is found

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B



Backward chaining

Idea: work backwards from the query q:

to prove q by BC,

check if q is known already, or

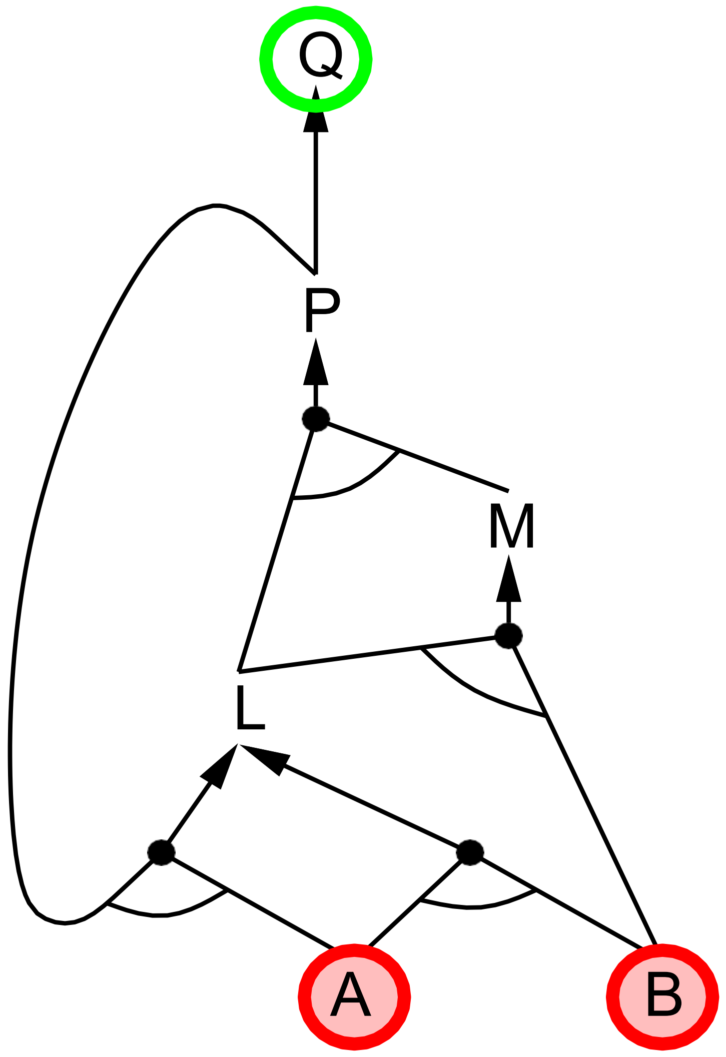
prove by BC all premises of some rule concluding q

Avoid loops: check if new subgoal is already on the goal stack

Avoid repeated work: check if new subgoal

1) has already been proved true, or
2) has already failed

$P \Rightarrow Q$
$L \wedge M \Rightarrow P$
$B \wedge L \Rightarrow M$
$A \wedge P \Rightarrow L$
$A \wedge B \Rightarrow L$
A
B



Backward chaining

Idea: work backwards

from the query q:

to prove q by BC,

check if q is known already, or

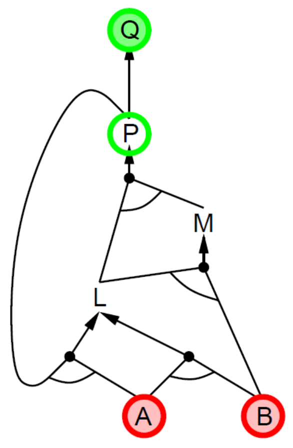
prove by BC all premises of some rule concluding q

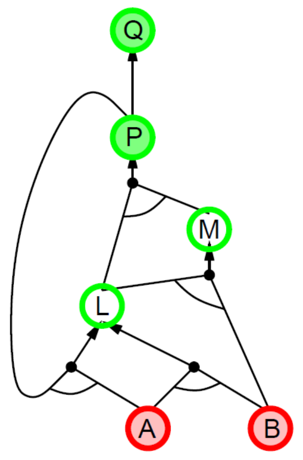
Avoid loops: is new subgoal on goal stack

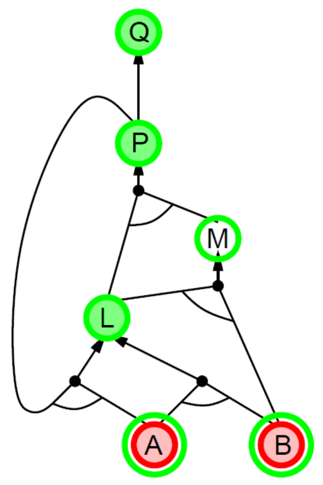
Avoid repeated work:

check if new subgoal

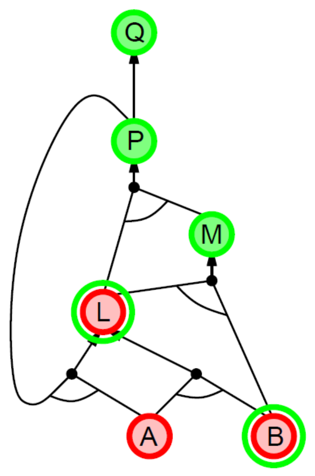
1) has already been proved true, or
2) has already failed

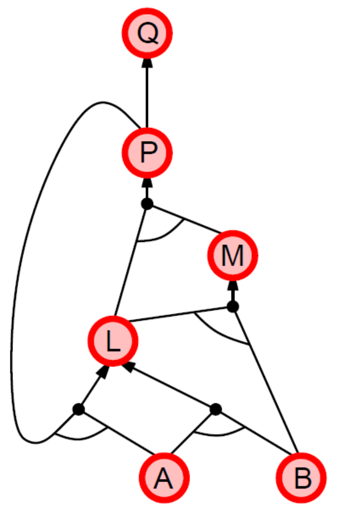






$$P \Rightarrow Q$$
$$L \wedge M \Rightarrow P$$
$$B \wedge L \Rightarrow M$$
$$A \wedge P \Rightarrow L$$
$$A \wedge B \Rightarrow L$$
$$A$$
$$B$$





Forward vs. backward chaining

FC is data-driven, cf. automatic, unconscious processing,

e.g., object recognition, routine decisions

May do lots of work that is irrelevant to the goal

BC is goal-driven, appropriate for problem-solving,

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

Complexity of BC can be much less than linear in size of KB

Resolution

Conjunctive Normal Form (CNF—universal)

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

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

Resolution inference rule (for CNF): complete for propositional logic

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

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 rule:

$$\frac{\begin{array}{l}\alpha \vee \beta \\ \neg\beta \vee \gamma\end{array}}{\alpha \vee \gamma}$$

Resolution is sound and complete for propositional logic

Conversion to CNF

$B_1 \Leftrightarrow (P_1 \vee P_2)$

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

   $(B_1 \Rightarrow (P_1 \vee P_2)) \wedge ((P_1 \vee P_2) \Rightarrow B_1)$

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

   $(\neg B_1 \vee P_1 \vee P_2) \wedge (\neg(P_1 \vee P_2) \vee B_1)$

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

   $(\neg B_1 \vee P_1 \vee P_2) \wedge ((\neg P_1 \wedge \neg P_2) \vee B_1)$

4. Apply distributivity law ($\vee$ over $\wedge$) and flatten: $A \vee (B \wedge C) \iff (A \vee B) \wedge (A \vee C)$

   $(\neg B_1 \vee P_1 \vee P_2) \wedge (\neg P_1 \vee B_1) \wedge (\neg P_2 \vee B_1)$

Resolution algorithm

To prove a.
Proof by contradiction, i.e., show $KB \wedge \neg a$ is unsatisfiable

If $KB \wedge \neg a$ is unsatisfiable then $KB \models a$

1. Add negation of query to KB
2. Pick two clauses with complementary literals that haven't been used before for resolution.
3. If there aren't any, then answer query is not entailed by KB.
4. Otherwise, compute resolvent and add it to KB,
5. If false in KB (contradiction found $KB \wedge \neg a$ unsatisfiable)

   Halt and answer the query IS entailed by KB
6. Else goto 2.

Resolution example

Given KB

i	A v B
ii	A -> C
iii	B -> C

We want to prove C

Does this KB ⊨ C

Step	Formula	Derivation
1	A v B	Given
2	¬A v C	Given ii
3	¬B v C	Given iii
4	¬C	Negation of C (To prove C)
5	B v C	1,2
6	¬A	2,4
7	¬B	3,4
8	C	5,7
9	Contradiction	4,8

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
- soundess: derivations produce only entailed sentences
- completeness: derivations can produce all entailed sentences

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

Propositional logic lacks expressive power

Pros and cons of propositional logic

- Propositional logic is declarative: pieces of syntax correspond to facts
- Propositional logic allows partial/disjunctive/negated information (unlike most data structures and databases)
- Propositional logic is compositional:
- meaning of $B_{1,1} \wedge P_{1,2}$ is derived from meaning of $B_{1,1}$ and of $P_{1,2}$
- Meaning in propositional logic is context-independent
- (unlike natural language, where meaning depends on context)
- Propositional logic has very limited expressive power (unlike natural language)
- E.g., cannot say "pits cause breezes in adjacent squares" except by writing one sentence for each square

First-order logic

Whereas propositional logic assumes world contains facts,
first-order logic (like natural language) assumes the world contains

- Objects: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
- Relations: 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, end of . . .

Logics in general

Language	Ontological Commitment	Epistemological Commitment
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 FOL: Basic elements

Constants	$KingJohn,\ \mathbf{2},\ UCB, \ldots$
Predicates	$Brother,\ >, \ldots$
Functions	$Sqrt,\ LeftLegOf, \ldots$
Variables	$x,\ y,\ a,\ b, \ldots$
Connectives	$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \Leftrightarrow$
Equality	$=$
Quantifiers	$\forall\ \exists$

Atomic and complex sentences

Atomic sentence = $predicate(term_1, \ldots, term_n)$
or $term_1 = term_2$

Term = $function(term_1, \ldots, term_n)$
or $constant$ or $variable$

E.g., $Brother(KingJohn, RichardTheLionheart)$
$> (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))$

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. $Sibling(KingJohn, Richard) \Rightarrow Sibling(Richard, KingJohn)$

Truth in first-order logic

Sentences are true with respect to a model and an interpretation

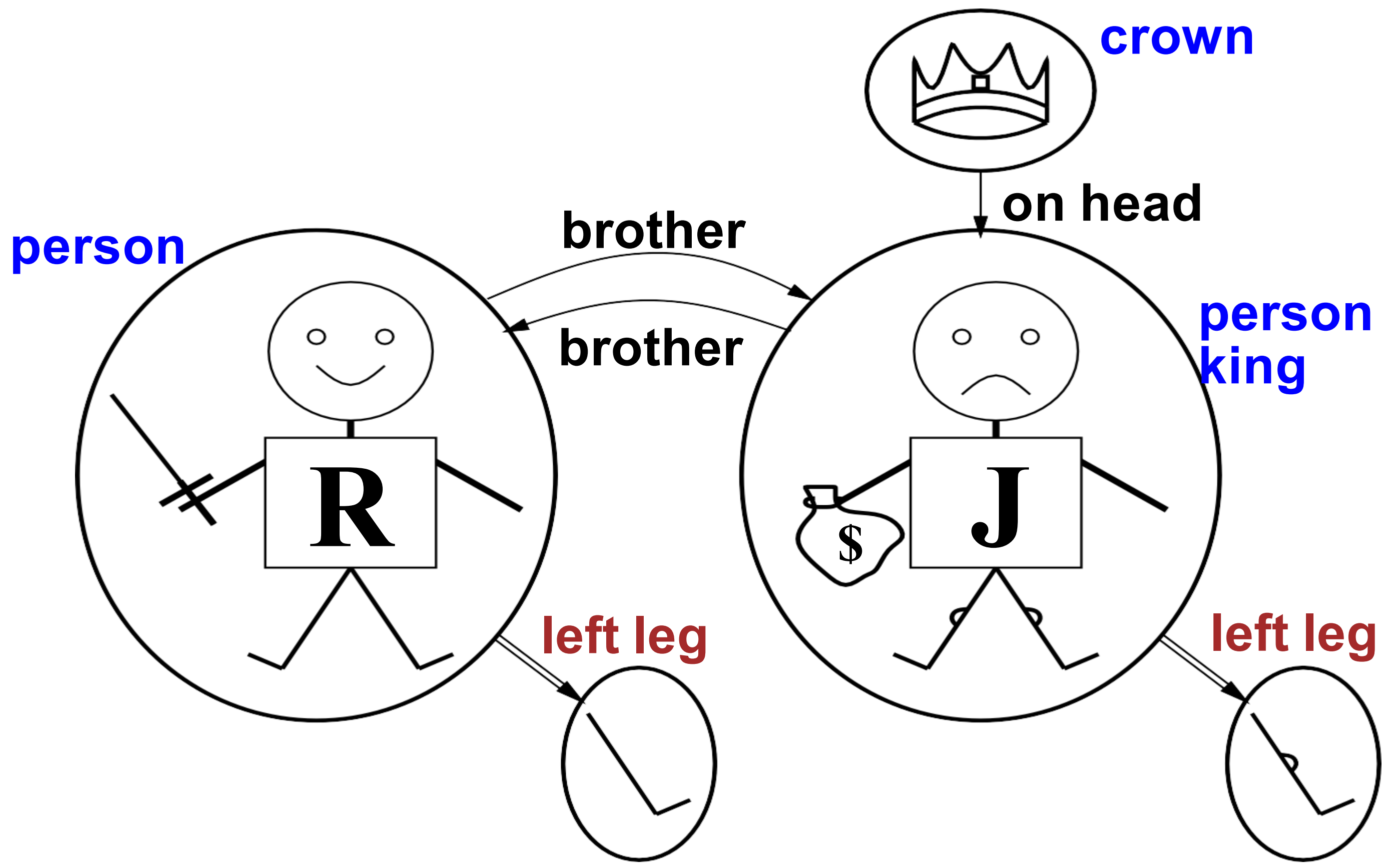
Model contains >= 1 objects (domain elements) and relations among them

Interpretation specifies referents for

constant symbols $\rightarrow$ objects
predicate symbols $\rightarrow$ relations
function symbols $\rightarrow$ functional relations

An atomic sentence $predicate(term_1, \ldots, term_n)$ is true iff
the objects referred to by $term_1, \ldots, term_n$
are in the relation referred to by $predicate$

Models for FOL: Example



Truth example

Consider the interpretation in which
Richard → Richard the Lionheart
John → the evil King John
Brother → the brotherhood relation

Under this interpretation, *Brother*(*Richard*, *John*) is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Figure 8.2 A model containing five objects, two binary relations (brother and on-head), three unary relations (person, king, and crown), and one unary function (left-leg).

Models for FOL: Lots!

Entailment in propositional logic can be computed by enumerating models

We can enumerate the FOL models for a given KB vocabulary:

```
For each number of domain elements $n$ from 1 to $\infty$
    For each $k$-ary predicate $P_k$ in the vocabulary
        For each possible $k$-ary relation on $n$ objects
            For each constant symbol $C$ in the vocabulary
                For each choice of referent for $C$ from $n$ objects . . .
```

Computing entailment by enumerating FOL models is not easy!

Universal quantification

$\forall\, \langle variables \rangle\ \langle sentence \rangle$

Everyone at Berkeley is smart:
$\forall x\ \ At(x, Berkeley)\ \Rightarrow\ Smart(x)$

$\forall x\ \ P$ is true in a model m iff P is true with x being each possible object in the model

Roughly speaking, equivalent to the conjunction of instantiations of P

$$\begin{aligned}
& (At(KingJohn, Berkeley)\ \Rightarrow\ Smart(KingJohn)) \\
\land\ & (At(Richard, Berkeley)\ \Rightarrow\ Smart(Richard)) \\
\land\ & (At(Berkeley, Berkeley)\ \Rightarrow\ Smart(Berkeley)) \\
\land\ & \ldots
\end{aligned}$$

Existential quantification

$\exists \langle variables \rangle \langle sentence \rangle$

Someone at Stanford is smart:
$\exists x \;\; At(x, Stanford) \land Smart(x)$

$\exists x \;\; P$ is true in a model m iff P is true with x being
some possible object in the model

Roughly speaking, equivalent to the disjunction of instantiations of P

$$
\begin{aligned}
& (At(KingJohn, Stanford) \land Smart(KingJohn)) \\
\lor \; & (At(Richard, Stanford) \land Smart(Richard)) \\
\lor \; & (At(Stanford, Stanford) \land Smart(Stanford)) \\
\lor \; & \ldots
\end{aligned}
$$

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

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

"Sibling" is symmetric

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

One's mother is one's female parent

$\forall x, y \;\; Mother(x, y) \;\Leftrightarrow\; (Female(x) \land Parent(x, y))$.

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

$\forall x, y \;\; FirstCousin(x, y) \;\Leftrightarrow\; \exists\, p, ps \;\; Parent(p, x) \land Sibling(ps, p) \land$
$Parent(ps, y)$

Equality

$term_1 = term_2$ is true under a given interpretation
if and only if $term_1$ and $term_2$ refer to the same object

E.g., $1 = 2$ and $\forall x \ (Sqrt(x) * Sqrt(x)) = x$ are satisfiable **(Discuss?)**
$2 = 2$ is valid

E.g., definition of (full) *Sibling* in terms of *Parent*:

$$\forall x, y \ \ Sibling(x, y) \ \Leftrightarrow \ [\neg(x = y) \land \exists m, f \ \ \neg(m = f) \land$$
$$Parent(m, x) \land Parent(f, x) \land Parent(m, y) \land Parent(f, y)]$$

Logic programming

Computation as inference on logical KBs

	Logic programming	Ordinary programming
1.	Identify problem	Identify problem
2.	Assemble information	Assemble information
3.	Tea break	Figure out solution
4.	Encode information in KB	Program solution
5.	Encode problem instance as facts	Encode problem instance as data
6.	Ask queries	Apply program to data
7.	Find false facts	Debug procedural errors

Should be easier to debug $Capital(NewYork, US)$ than $x := x + 2$!

Prolog systems

Basis: backward chaining with Horn clauses + bells & whistles Widely used in Europe, Japan (basis of 5th Generation project) Compilation techniques ⇒ approaching a billion LIPS

Program = set of clauses = head :- literal_1, … literal_n.

```
trades(X) :- person(X), item(Y), sells(X,Y,Z), has_relation(X,Z).
```

Efficient unification by finding matching values for Variables X,Y,Z.
Efficient retrieval of matching clauses by direct linking Depth-first, left-to-right backward chaining
Built-in predicates for arithmetic etc., e.g., X is Y*Z+3
Closed-world assumption ("negation as failure") e.g., given

```
alive(X)          :- not dead(X). alive(joe)
```

succeeds if dead(joe) fails

Prolog examples

- Depth-first search from a start state X:

```
dfs(X) :-        goal(X).
dfs(X) :-        successor(X,S),dfs(S).
```

- No need to loop over S: successor succeeds for each

- Appending two lists to produce a third:

```
append([],Y,Y).
append([X|L],Y,[X|Z])  :-      append(L,Y,Z).
```

- query: append(A,B,[1,2]) ?

```
answers: A=[]     B=[1,2]
         A=[1]    B=[2]
         A=[1,2]  B=[]
```

More examples

- SWI Prolog https://www.swi-prolog.org/
- SWISH -- examples.swinb (swi-prolog.org)
- Download and install for free
- Also available on appsanywhere
- Try the samples.

- Many books.
  - Prolog Programming for Artificial Intelligence, Ivan Bratko
  - Programming in Prolog, Clocksin and Mellish

Questions and Comments?