Artificial Intelligence

Propositional Logic

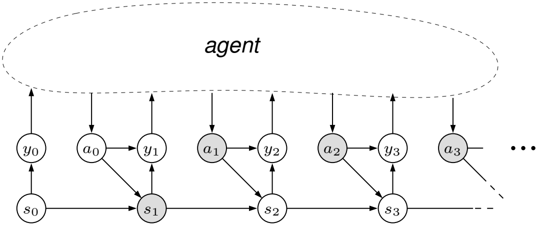
Marc Toussaint
University of Stuttgart
Winter 2015/16

(slides based on Stuart Russell's AI course)

Outline

- Knowledge-based agents
- Wumpus world
- Logic in general—models and entailment
- Propositional (Boolean) logic
- Equivalence, validity, satisfiability
- Inference rules and theorem proving
    - forward chaining
    - backward chaining
    - resolution

Knowledge bases



- An agent maintains a knowledge base

  Knowledge base = set of sentences of a *formal* language

Wumpus World description

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

Squares adjacent to wumpus are smelly

Squares adjacent to pit are breezy

Glitter iff gold is in the same square

Shooting kills wumpus if you are facing it

The wumpus kills you if in the same square

Shooting uses up the only arrow

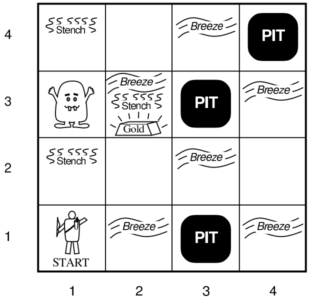
Grabbing picks up gold if in same square

Releasing drops the gold in same square

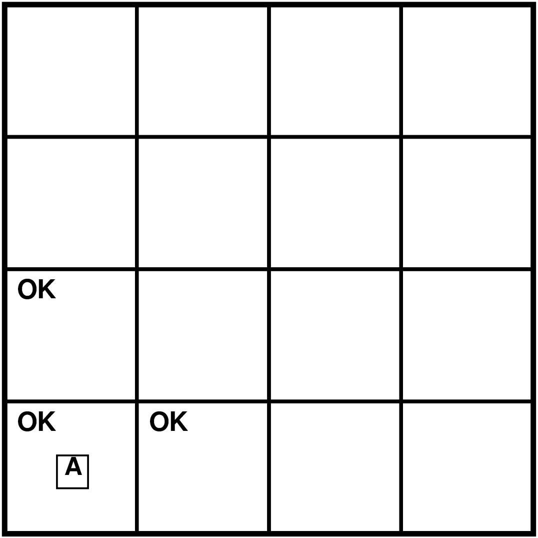
Actuators Left turn, Right turn,

Forward, Grab, Release, Shoot, Climb

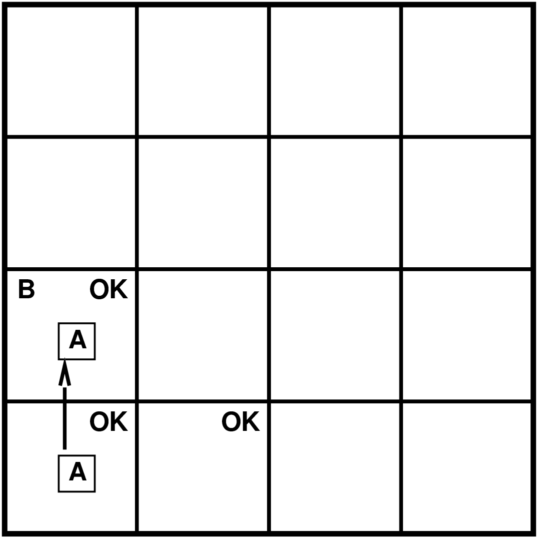
Sensors Breeze, Glitter, Stench, Bump, Scream



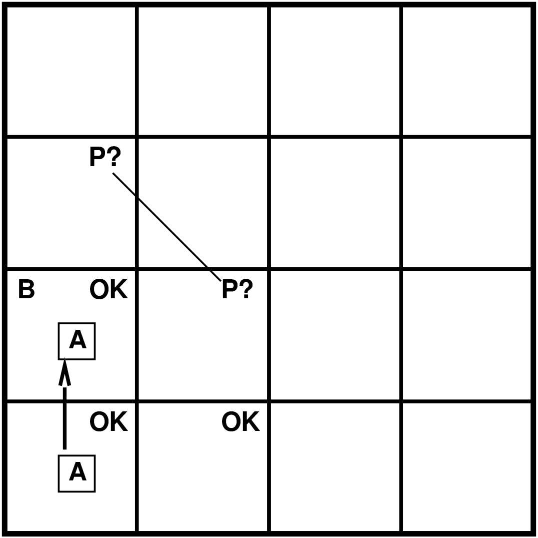
Exploring a wumpus world



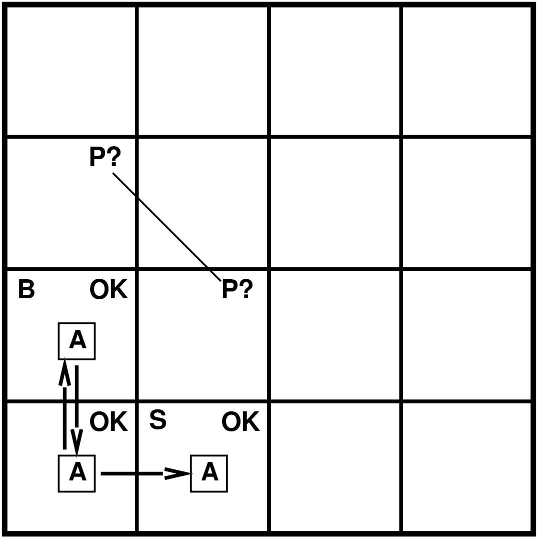
Exploring a wumpus world



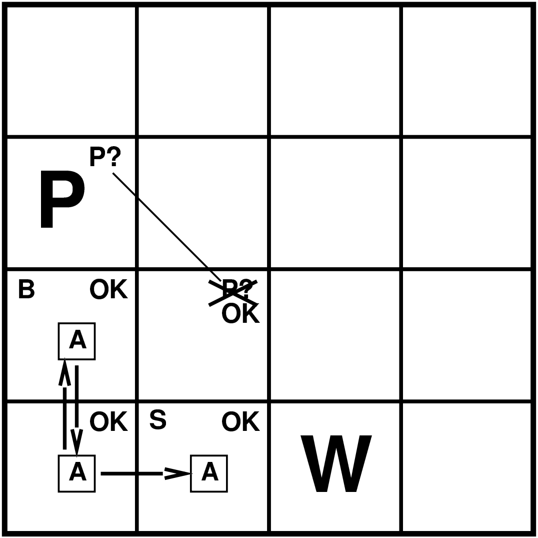
Exploring a wumpus world



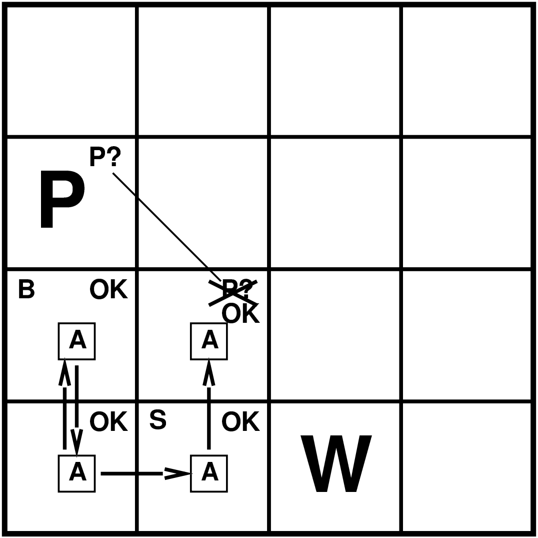
Exploring a wumpus world



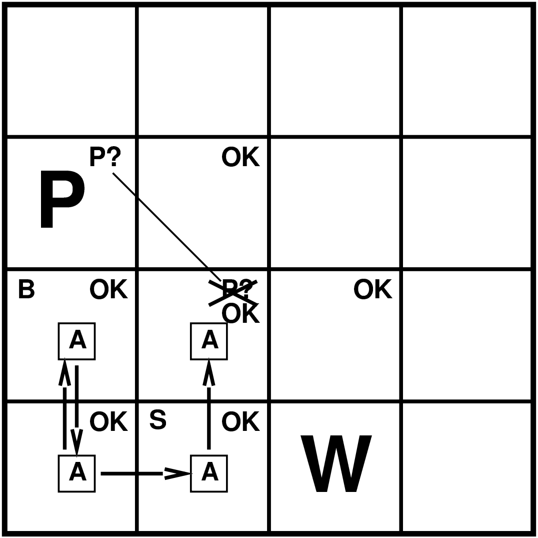
Exploring a wumpus world



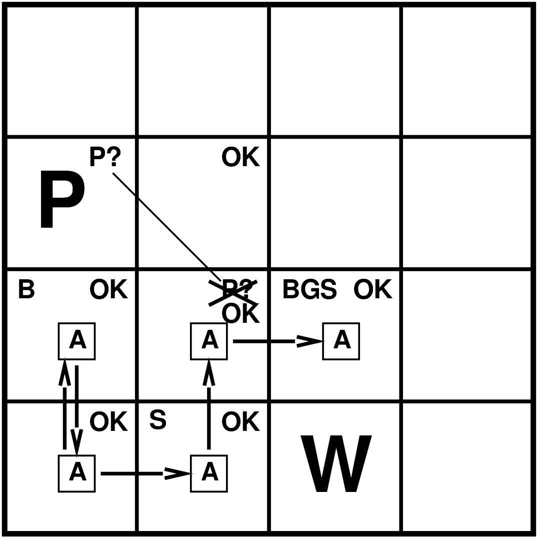
Exploring a wumpus world



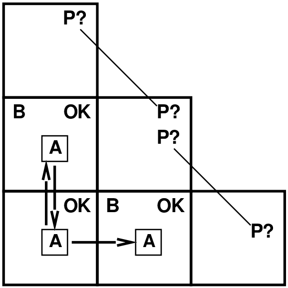
Exploring a wumpus world



Exploring a wumpus world

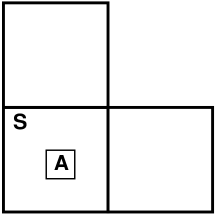


Other tight spots



Breeze in (1,2) and (2,1)
$\Rightarrow$ no safe actions

Assuming pits uniformly distributed, (2,2) has pit w/ prob 0.86, vs. 0.31



Smell in (1,1) $\Rightarrow$ cannot move
Can use a strategy of coercion:
shoot straight ahead
wumpus was there $\Rightarrow$ dead $\Rightarrow$ safe
wumpus wasn't there $\Rightarrow$ safe

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:

$$KB \models \alpha$$

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 Giants won” and “the Reds won” entails “Either the Giants won or the Reds won”

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

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

Models

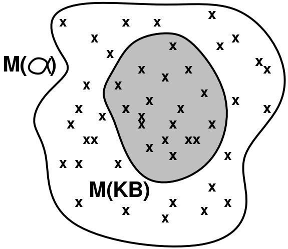
Given a logical sentence, when is its truth uniquely defined in a world?

Logicians typically think in terms of models, which are formally structured worlds
(e.g., full abstract description of a world, configuration of all variables, world state)
with respect to which truth can uniquely be evaluated

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 $M(KB) \subseteq M(\alpha)$



E.g. KB = Giants won and Reds won

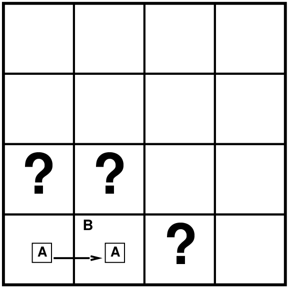
α = Giants won

Entailment in the wumpus world

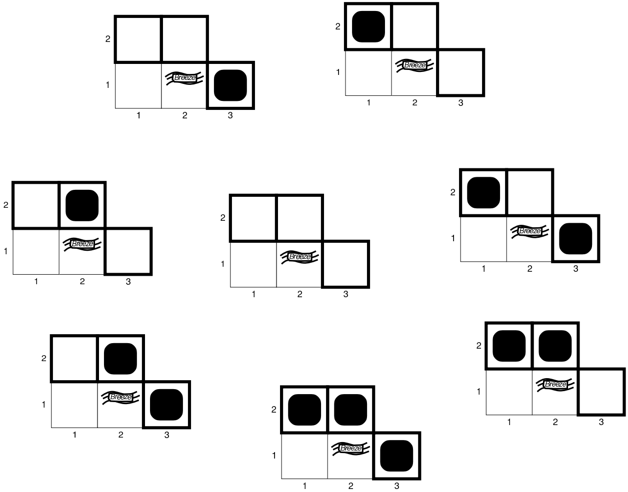
Situation after detecting nothing in [1,1],
moving right, breeze in [2,1]

Consider possible models for ?s
assuming only pits

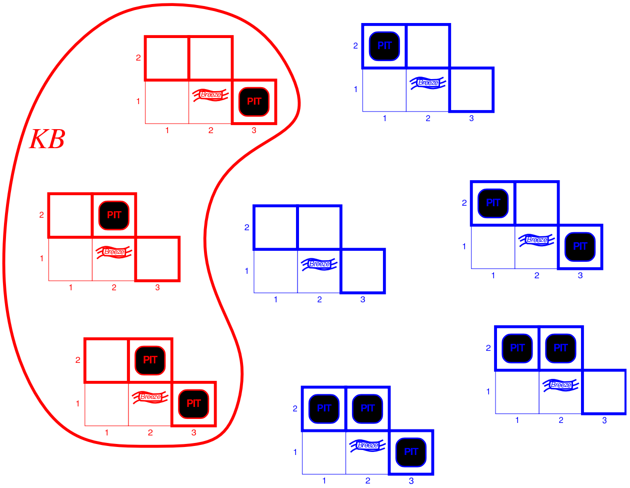
3 Boolean choices $\Rightarrow$ 8 possible models



Wumpus models

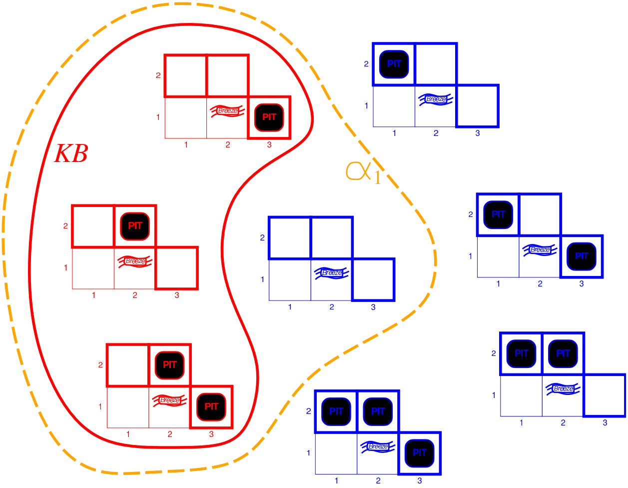


Wumpus models



KB = wumpus-world rules + observations

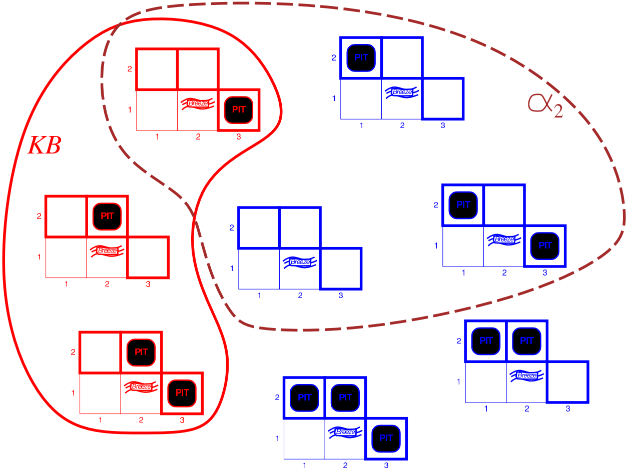
Wumpus models



KB = wumpus-world rules + observations

α_1 = "[1,2] is safe", $KB \models \alpha_1$, proved by model checking

Wumpus models



KB = wumpus-world rules + observations

α_2 = "[2,2] is safe", $KB \not\models \alpha_2$

Inference

Inference in the general sense means: Given some pieces of information (prior, observed variabes, knowledge base) what is the implication (the implied information, the posterior) on other things (non-observed variables, sentence)

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i
Consequences of KB are a haystack; α is a needle.
Entailment = needle in haystack; inference = finding it
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 $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which 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.

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 grammar

$$
\begin{array}{lcl}
\langle\text{sentence}\rangle & \rightarrow & \langle\text{atomic sentence}\rangle \mid \langle\text{complex sentence}\rangle \\
\langle\text{atomic sentence}\rangle & \rightarrow & \text{true} \mid \text{false} \mid P \mid Q \mid R \mid \ldots \\
\langle\text{complex sentence}\rangle & \rightarrow & \neg\, \langle\text{sentence}\rangle \\
 & & \mid (\langle\text{sentence}\rangle \wedge \langle\text{sentence}\rangle) \\
 & & \mid (\langle\text{sentence}\rangle \vee \langle\text{sentence}\rangle) \\
 & & \mid (\langle\text{sentence}\rangle \;\Rightarrow\; \langle\text{sentence}\rangle) \\
 & & \mid (\langle\text{sentence}\rangle \;\Leftrightarrow\; \langle\text{sentence}\rangle)
\end{array}
$$

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	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.,

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

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*

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$\neg P_{1,1}$$

$$\neg B_{1,1}$$

$$B_{2,1}$$

"Pits cause breezes in adjacent squares"

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i,j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i,j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

"Pits cause breezes in adjacent squares"

$$\begin{aligned} B_{1,1} &\quad\Leftrightarrow\quad (P_{1,2} \vee P_{2,1}) \\ B_{2,1} &\quad\Leftrightarrow\quad (P_{1,1} \vee P_{2,2} \vee P_{3,1}) \end{aligned}$$

"A square is breezy *if and only if* there is an adjacent pit"

Truth tables for inference

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	R_1	R_2	R_3	R_4	R_5	KB
false	*false*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*false*	*false*
false	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*false*	*true*	*false*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*true*	*true*	*false*	*true*	*true*	*false*
false	*true*	*false*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*false*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*false*	*true*	*true*	*true*	*true*	*true*	*true*	*true*	*true*
false	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*false*	*false*	*true*	*true*	*false*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*true*	*true*	*false*	*true*	*false*

Enumerate rows (different assignments to symbols),
if KB is true in row, check that α is too

Inference by enumeration

Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [ ])
--------------------------------------------------------------------
function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true
   else do
      P ← FIRST(symbols); rest ← REST(symbols)
      return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
             TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

$O(2^n)$ for n symbols

Logical equivalence

Two sentences are logically equivalent iff true in same models:

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

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

Validity and satisfiability

A sentence is valid if it is true in *all* models,

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

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., $A \vee B$, 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

i.e., prove α by *reductio ad absurdum*

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 as operators in a standard search alg.

– Typically require translation of sentences into a normal form

Model checking

truth table enumeration (always exponential in n)

improved backtracking, e.g., Davis–Putnam–Logemann–Loveland (see book)

heuristic search in model space (sound but incomplete)

e.g., min-conflicts-like hill-climbing algorithms

Forward and backward chaining

Applicable when KB is in Horn Form

Horn Form (restricted)

KB = *conjunction* of *Horn clauses*

Horn clause =

– proposition symbol; or

– (conjunction of symbols) $\Rightarrow$ symbol

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

Modus Ponens (for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Can be used with forward chaining or backward chaining.

These algorithms are very natural and run in *linear* time

Forward chaining

Idea: fire any rule whose premises are satisfied in the KB,

add its conclusion to the KB, until query is found

$$P \Rightarrow Q$$

$$L \wedge M \Rightarrow P$$

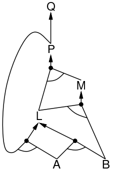
$$B \wedge L \Rightarrow M$$

$$A \wedge P \Rightarrow L$$

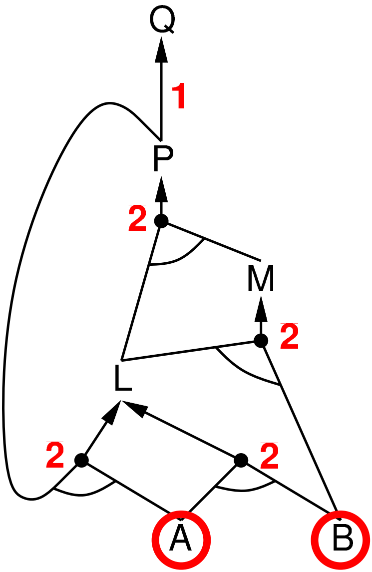
$$A \wedge B \Rightarrow L$$

$$A$$

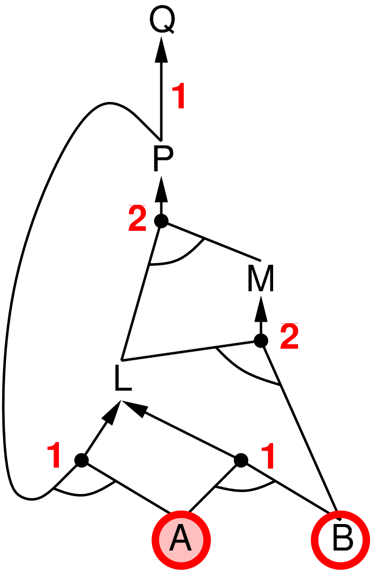
$$B$$



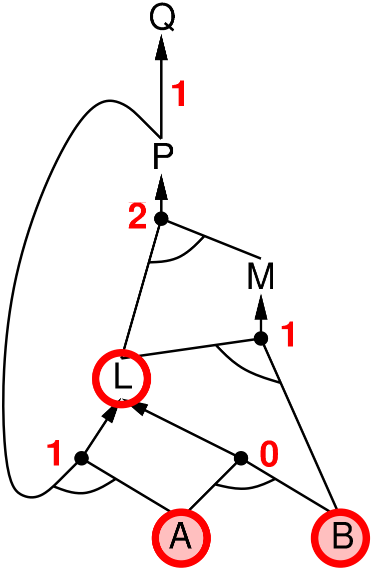
Forward chaining example



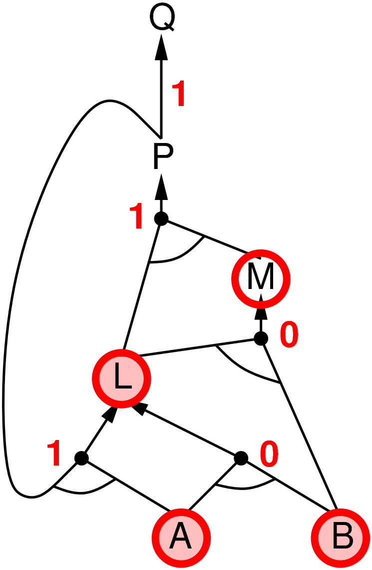
Forward chaining example



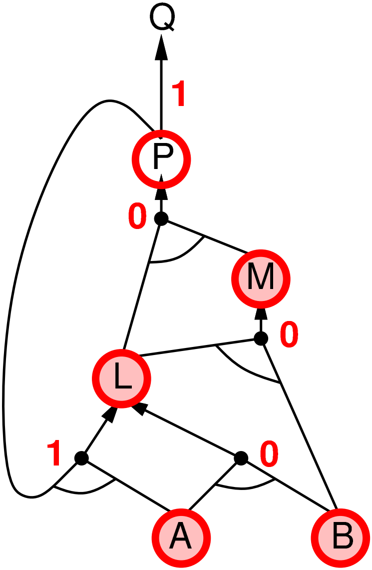
Forward chaining example



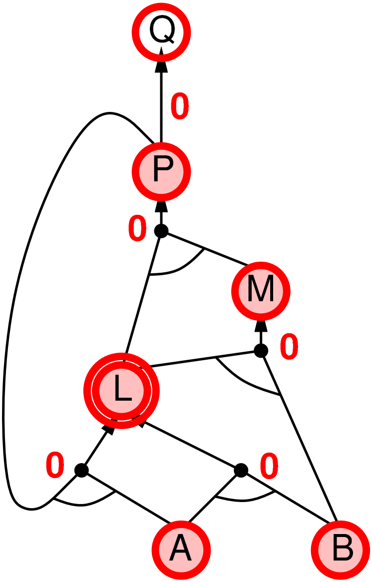
Forward chaining example



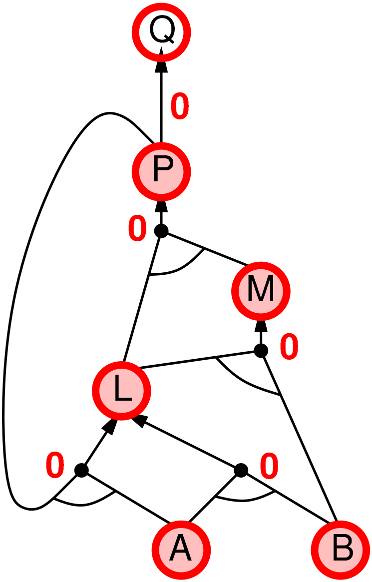
Forward chaining example



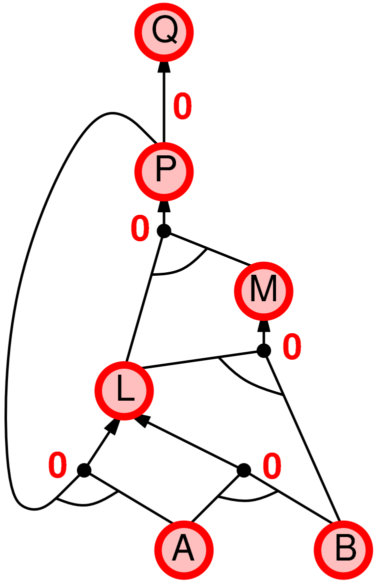
Forward chaining example



Forward chaining example



Forward chaining example



Forward chaining algorithm

```
function PL-FC-ENTAILS?(KB, q) returns true or false
   inputs: KB, the knowledge base, a set of propositional Horn clauses
           q, the query, a proposition symbol
   local variables: count, a table, indexed by clause, initially the number of premises
                    inferred, a table, indexed by symbol, each entry initially false
                    agenda, a list of symbols, initially the symbols known in KB

   while agenda is not empty do
      p ← POP(agenda)
      unless inferred[p] do
         inferred[p] ← true
         for each Horn clause c in whose premise p appears do
            decrement count[c]
            if count[c] = 0 then do
               if HEAD[c] = q then return true
               PUSH(HEAD[c], agenda)
   return false
```

Proof of completeness

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

   Proof: Suppose a clause $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ is false in m

   Then $a_1 \wedge \ldots \wedge a_k$ is true in m and b is false in m

   Therefore the algorithm has not reached a fixed point!
4. Hence m is a model of KB
5. If $KB \models q$, q is true in *every* model of KB, including m

General idea: construct any model of KB by sound inference, check α

Backward chaining

Idea: work backwards from the query q:

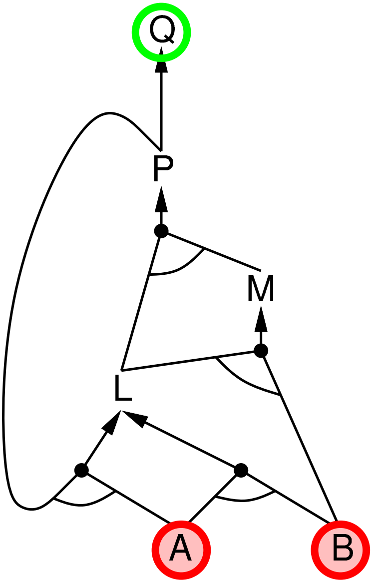
- to prove q by BC,
  - check if q is known already, or
  - prove by BC all premises of some rule concluding q

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

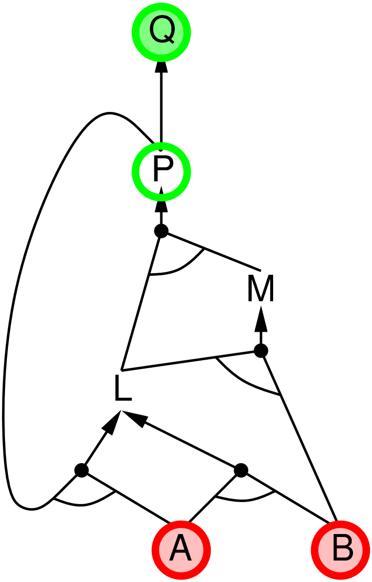
Avoid repeated work: check if new subgoal

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

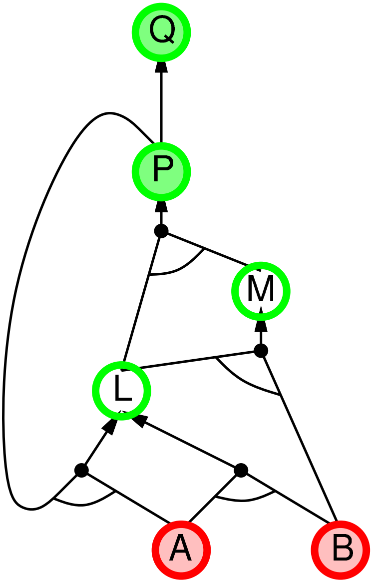
Backward chaining example



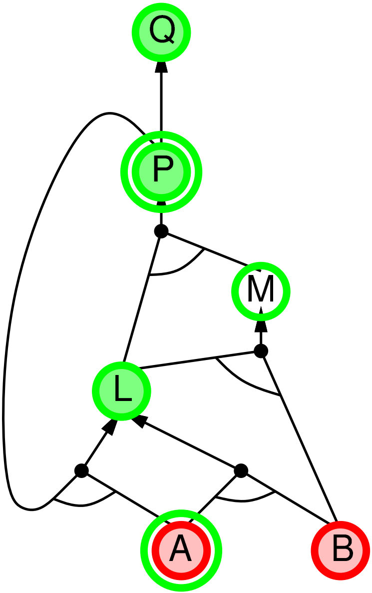
Backward chaining example



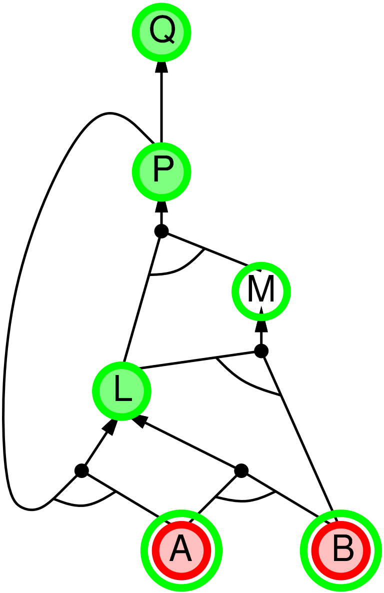
Backward chaining example



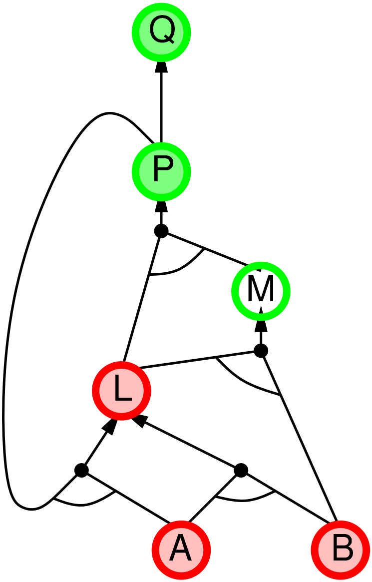
Backward chaining example



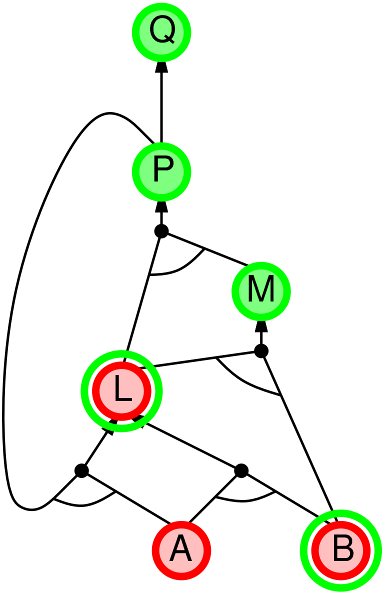
Backward chaining example



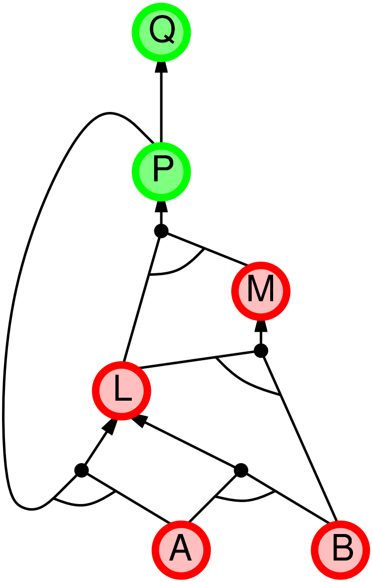
Backward chaining example



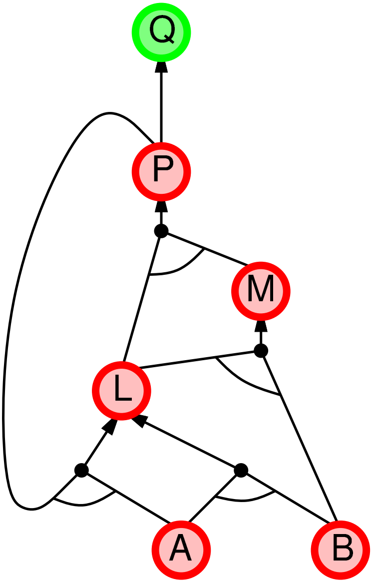
Backward chaining example



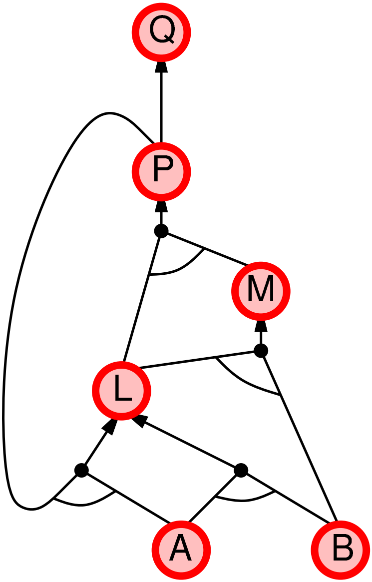
Backward chaining example



Backward chaining example



Backward chaining example



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)

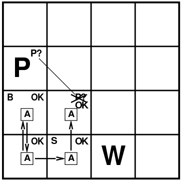
$$\textit{conjunction}\text{ of }\underbrace{\textit{disjunctions}\text{ of }\textit{literals}}_{\textit{clauses}}$$

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 is sound and complete for propositional logic

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 algorithm

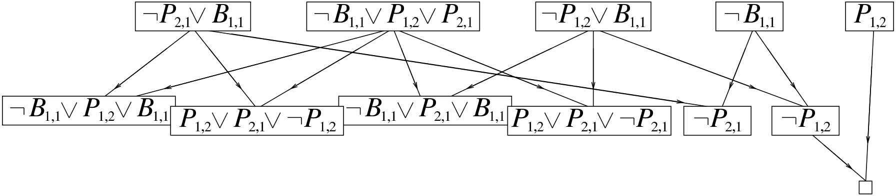
Proof by contradiction, i.e., show $KB \wedge \neg\alpha$ unsatisfiable

```
function PL-RESOLUTION(KB, α) returns true or false
   inputs: KB, the knowledge base, a sentence in propositional logic
           α, the query, a sentence in propositional logic

   clauses ← the set of clauses in the CNF representation of KB ∧ ¬α
   new ← { }
   loop do
       for each C_i, C_j in clauses do
           resolvents ← PL-RESOLVE(C_i, C_j)
           if resolvents contains the empty clause then return true
           new ← new ∪ resolvents
       if new ⊆ clauses then return false
       clauses ← clauses ∪ new
```

Resolution example

$$KB = (B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})) \wedge \neg B_{1,1} \qquad \alpha = \neg P_{1,2}$$



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

Wumpus world requires the ability to represent partial and negated information, reason by cases, etc.

Forward, backward chaining are linear-time, complete for Horn clauses

Resolution is complete for propositional logic

Propositional logic lacks expressive power

Dictionary: logic in general

a logic: a language, elements α are sentences, (grammar example: slide 34)

model m: a world/state description that allows us to evaluate $\alpha(m) \in \{\mathsf{true}, \mathsf{false}\}$ uniquely for any sentence α, $M(\alpha) = \{m : \alpha(m) = \mathsf{true}\}$

entailment $\alpha \models \beta$: $M(\alpha) \subseteq M(\beta)$, "$\forall_m : \alpha(m) \Rightarrow \beta(m)$" (Folgerung)

equivalence $\alpha \equiv \beta$: iff ($\alpha \models \beta$ and $\beta \models \alpha$)

KB: a set of sentences

inference procedure i can infer α from KB: $KB \vdash_i \alpha$

soundness of i: $KB \vdash_i \alpha$ implies $KB \models \alpha$ (Korrektheit)

completeness of i: $KB \models \alpha$ implies $KB \vdash_i \alpha$

Dictionary: propositional logic

conjunction: $\alpha \wedge \beta$, disjunction: $\alpha \vee \beta$, negation: $\neg\alpha$

implication: $\alpha \Rightarrow \beta \ \equiv \ \neg\alpha \vee \beta$, biconditional: $\alpha \Leftrightarrow \beta \ \equiv \ (\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$

Note: $\models$ and $\equiv$ are statements about sentences in a logic; $\Rightarrow$ and $\Leftrightarrow$ are symbols in the grammar of propositional logic

α valid: true for *any* model, e.g.: $KB \models \alpha$ iff $[(KB \Rightarrow \alpha)$ is valid] (allgemeingültig)

α unsatisfiable: true for *no* model, e.g.: $KB \models \alpha$ iff $[(KB \wedge \neg\alpha)$ is unsatisfiable]

literal: A or $\neg A$, clause: disjunction of literals, CNF: conjunction of clauses

Horn clause: symbol | (conjunction of symbols $\Rightarrow$ symbol), Horn form: conjunction of Horn clauses

Modus Ponens rule: complete for Horn KBs $\frac{\alpha_1,\ldots,\alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$

Resolution rule: complete for propositional logic in CNF, let "$\ell_i = \neg m_j$":

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