TDDC17

Seminar 5 (and 6)

Ch. 7
Knowledge Representation I
Logical Agents
Intuitions
Propositional Logic
Propositional Theorem Proving:
DPLL
(Resolution Theorem Proving)

Some additional help: (click “literature” on the IDA course web page)
https://www.ida.liu.se/~TDDC17/info/literature/szalas-cugs-lectures.pdf

Or on the LISAM Documents page.

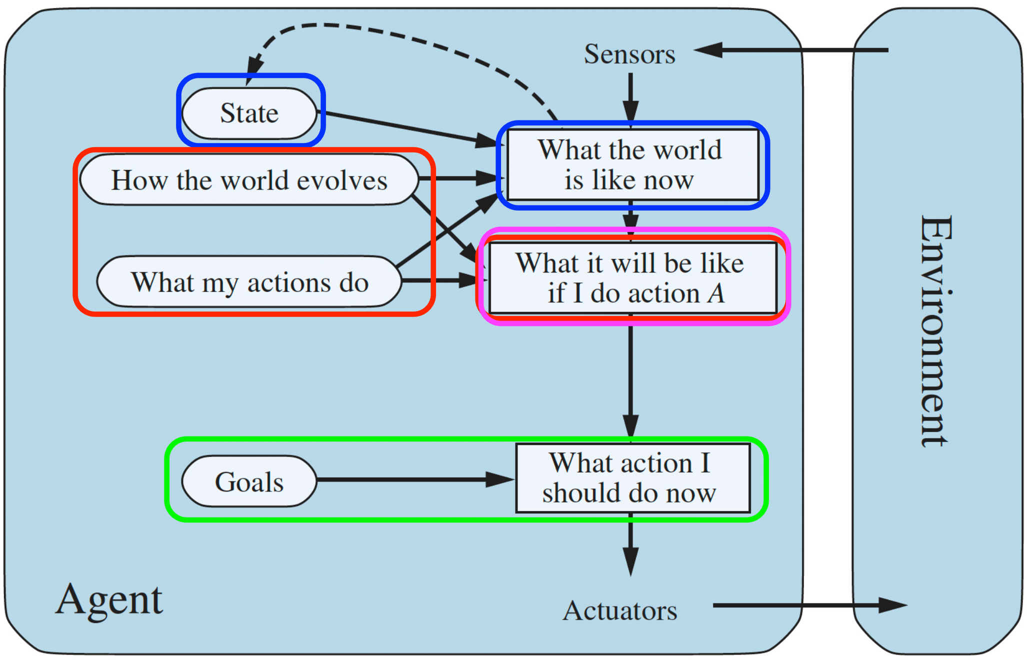
Patrick Doherty
Dept of Computer and Information Science
Artificial Intelligence and Integrated Computer Systems Division
LINKÖPING UNIVERSITY

Model-based, Goal-Directed Agents

Keeps track of world-states

Has knowledge about the ways of the world

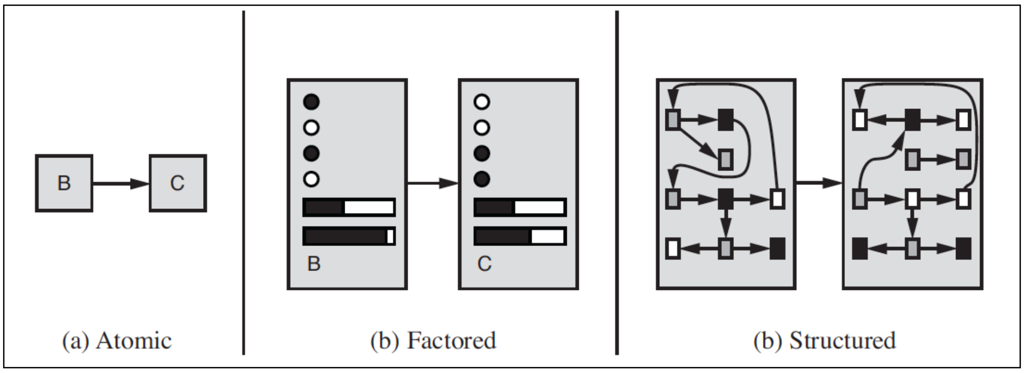
It is goal directed



Anticipates by internal simulation/inference

LINKÖPING UNIVERSITY

Representing States/Knowledge



(a) Atomic (b) Factored (b) Structured

<u>So far:</u>

Uninformed search

Heuristic search

Constraints

<u>Today/Next Seminar:</u>

Propositional Logic

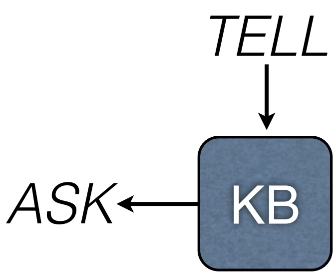
1st-Order Logic

Answer Set Programs

Generic Model-based Agent

function KB-AGENT(*percept*) **returns** an *action*
  persistent: *KB*, a knowledge base
    t, a counter, initially 0, indicating time

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, t))
  action ← ASK(*KB*, MAKE-ACTION-QUERY(t))
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, t))
  $t \leftarrow t + 1$
  return *action*



Advice Taker [McCarthy 1958]

Programs with common sense

Observe → Internalize Observation

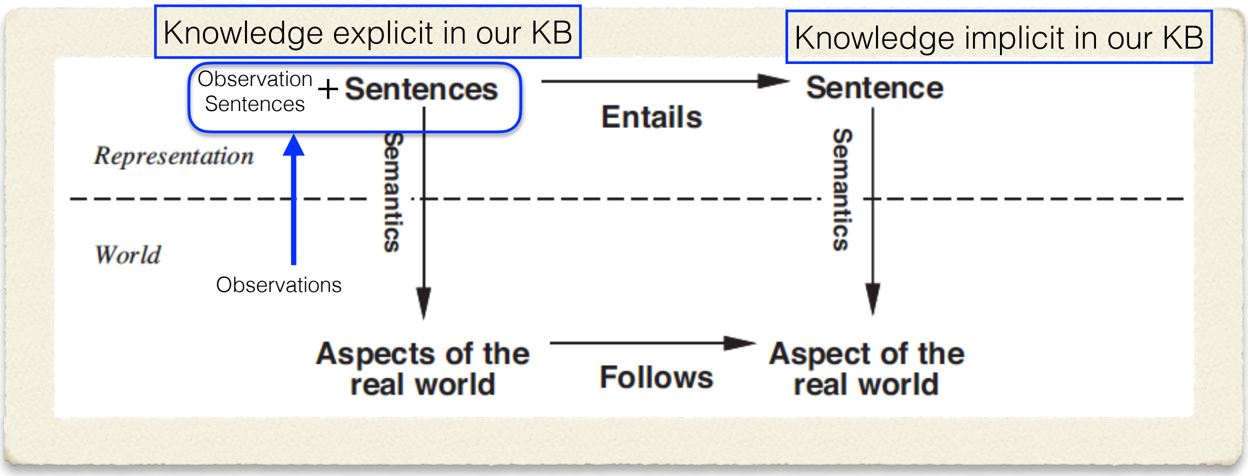
ASK for next action → Reason about what is best

Make a Decision → TELL what action is decided

Execute Action → Loop

Declarative Approach: Specify "What", not "How"!

Knowledge Representation and Logic



What is our representation language?
How is it grounded causally in the world?

Truth preservation (soundness) guarantees fidelity of entailments to the world under the assumption that observation sentences (sensing) are correct, in addition to background knowledge in the KB.

LINKÖPING UNIVERSITY

Knowledge Representation Hypothesis

Characterizes our assumptions about such systems

Any mechanically embodied intelligent process will be comprised of structural ingredients that

a) we as external observers naturally take to represent a propositional account of the knowledge that the overall process exhibits, and

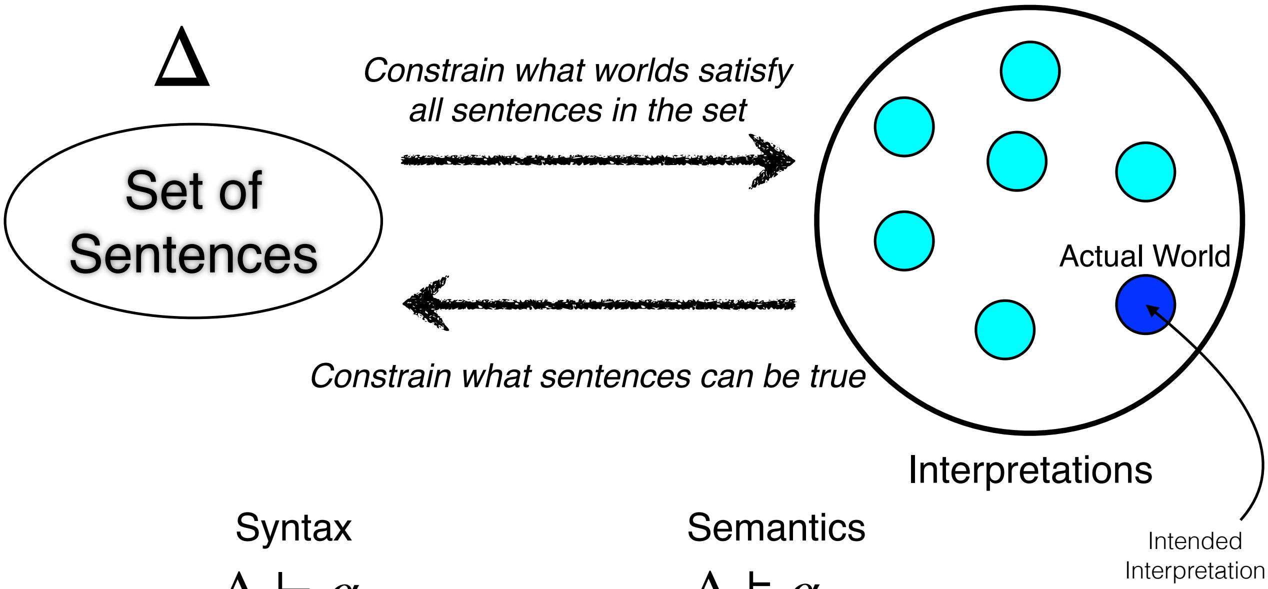
b) independent of such external semantical attribution, play a formal but causal and essential role in engendering the behavior that manifests that knowledge. [Brian Smith, 1982]

Recall the Physical Symbol System hypothesis!

LINKÖPING UNIVERSITY

One useful perspective: Knowledge as Constraints!

Knowledge Base | **Possible Worlds**



Syntax	Semantics
$\Delta \vdash \alpha$	$\Delta \models \alpha$
Inference	Entailment

LINKÖPING UNIVERSITY

Logic as a Representation Language

What is Logic? | Logic is about Reasoning | Logic is about Thought

Given a set of facts Δ taken to hold as true about the "world" and given an assertion α about the "world", is there a good argument for believing that α holds based on the initial set of facts Δ?

> Logic in the general sense is about making distinctions between good arguments and bad arguments and the different criteria that may be used in making this distinction.
> Deduction is one such criteria. (There are others!)

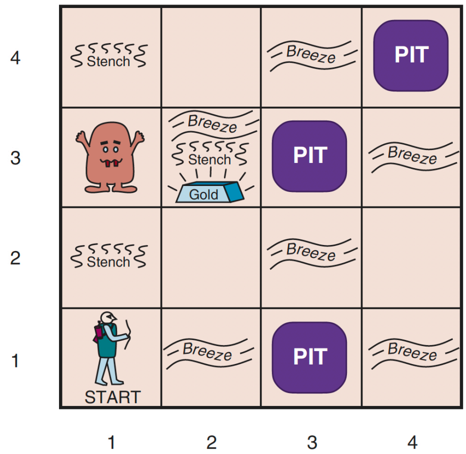
> Logic in the more restricted sense is about the study of mathematical theories for formalizing the distinction between good/bad arguments and mechanizing ways to make these distinctions

LINKÖPING UNIVERSITY

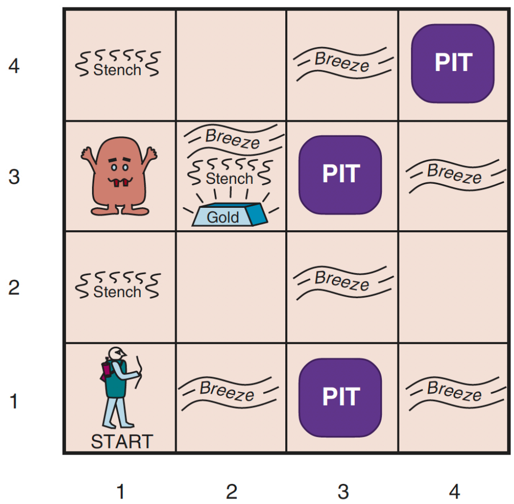
Wumpus World

The Wumpus World is a cave consisting of rooms connected by passageways. Lurking somewhere in the cave is a Wumpus, a beast that eats anyone who enters its room. The Wumpus can be shot by an agent, but the agent only has one arrow. Some rooms contain bottomless pits that will trap anyone who wanders into such a room. There is also the possibility of finding a heap of gold.

This is the goal of anyone who enters the Wumpus World. Find the Gold and bring it back to the start cell!



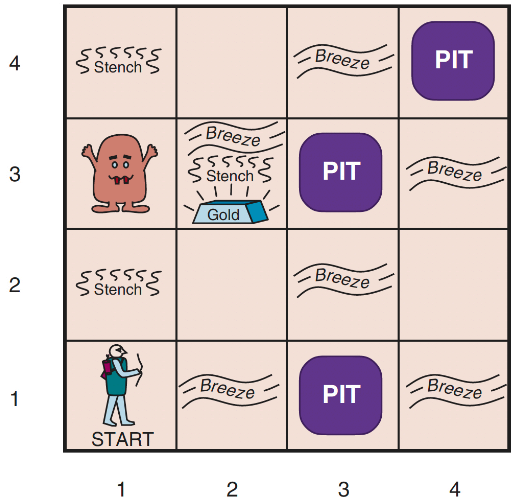
The Task Environment



<u>Performance Measure</u>
- +1000 for picking up gold,
- -1000 for falling into a pit or being eaten by a Wumpus,
- -1 for each action taken, and
- -10 for using an arrow.

<u>Environment</u>
4x4 grid of rooms. Square [1,1] is initial state with agent facing to the right. Locations of gold, and wumpus are chosen randomly, with a uniform distribution, from all squares but [1,1].
Each square other than [1,1] can contain a pit with probability 0.2.

The Task Environment



Actuators

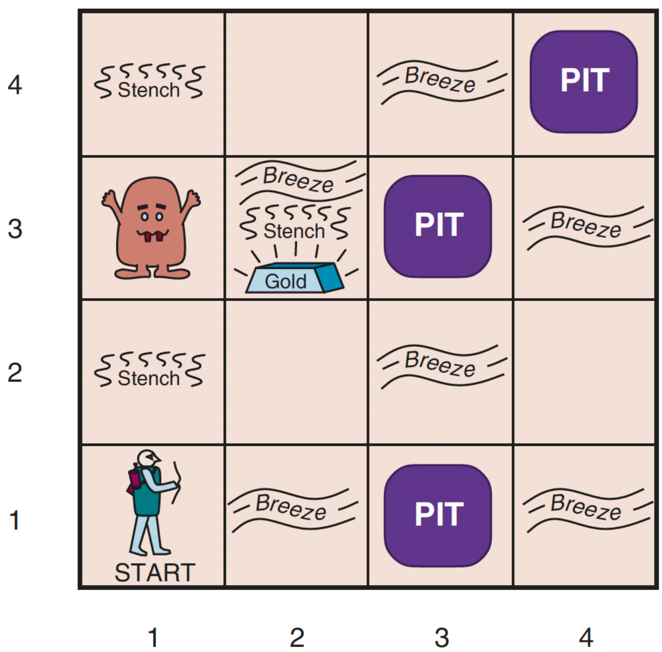
- The agent can **Move forward**, **Turn right** or **Turn left** by 90 degrees
- **Grab** can be used to pick up an object in the same square as the agent.
- **Shoot** can be used to shoot the single arrow in a straight line until it hits something (Wumpus or a boundary wall)

Sensors

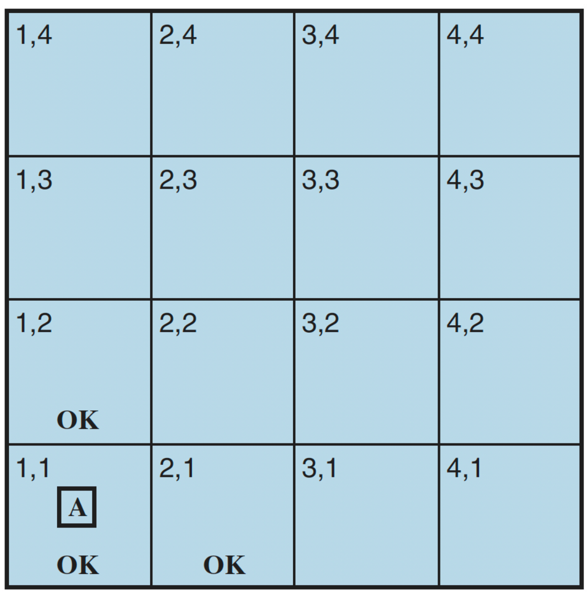
- A **stench** is perceived in the square containing a Wumpus or in those directly adjacent (not diagonal) to the Wumpus
- A **breeze** is perceived in a square directly adjacent to a pit
- A **glitter** is perceived in a square with gold in it.
- A **bump** is perceived if an agent walks into a wall.
- When the wumpus dies it emits a horrible **scream**.

An Example: Wumpus World

Reality



Agent A's View



Let's Explore through Reasoning!

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2	3,2	4,2
1,1 $\boxed{A}$ **OK**	2,1 **OK**	3,1	4,1

In $Rm_{1,1}$, there is no breeze or stench:

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

Consequently, $Rm_{2,1}$ and $Rm_{1,2}$ are safe:

$$OK_{2,1} \wedge OK_{1,2}$$

KB:

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$

LINKÖPING UNIVERSITY

13

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

A moves to $Rm_{2,1}$ and feels a breeze: $B_{2,1}$

KB

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}$

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 **OK**	2,2 **P?**	3,2	4,2
1,1 **V** **OK**	2,1 $\boxed{A}$ **B** **OK**	3,1 **P?**	4,1

What can **A** conclude about pits in its vicinity?

Given , $B_{2,1}$ there may be a Pit in either $Rm_{2,2}$ or $Rm_{3,1}$: $P_{2,2} \vee P_{3,1}$

KB

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1} \quad P_{2,2} \vee P_{3,1}$

Partial Observability as disjunctive information

14

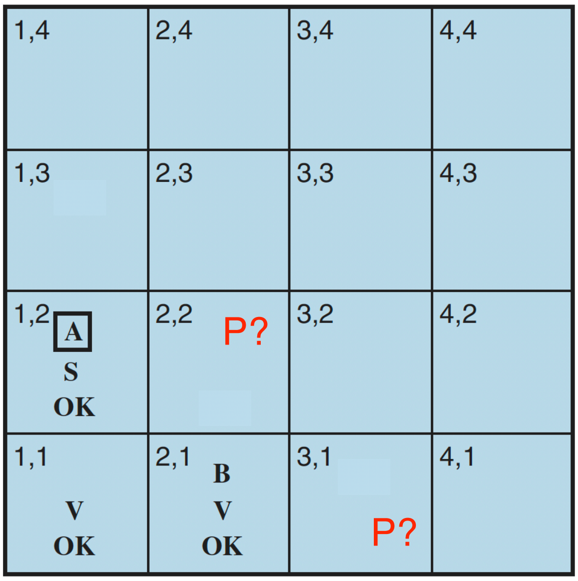
- A = *Agent*
- B = *Breeze*
- G = *Glitter, Gold*
- OK = *Safe square*
- P = *Pit*
- S = *Stench*
- V = *Visited*
- W = *Wumpus*

Since there may be a Pit in either $Rm_{2,2}$ or $Rm_{3,1}$:

$$P_{2,2} \vee P_{3,1}$$

A decides to move back to $Rm_{1,1}$ and then to $Rm_{1,2}$.

A then senses a stench in $Rm_{1,2}$: $S_{1,2}$



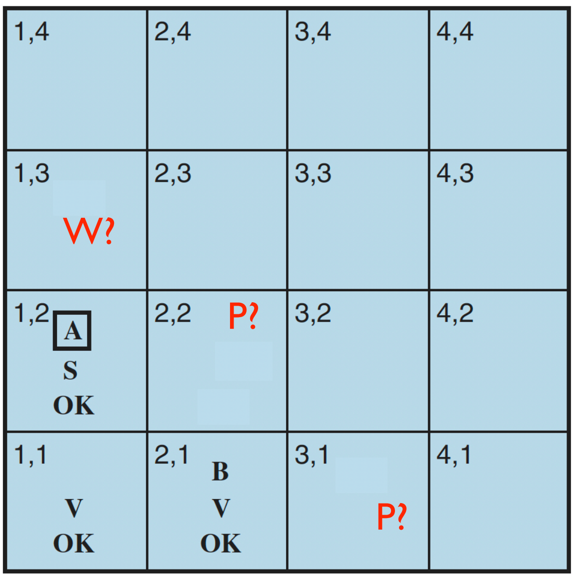
$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}$ **KB**

What can A infer about the Wumpus and Pits in the vicinity?

15

- A = *Agent*
- B = *Breeze*
- G = *Glitter, Gold*
- OK = *Safe square*
- P = *Pit*
- S = *Stench*
- V = *Visited*
- W = *Wumpus*

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}$ **KB**



Given $S_{1,2}$, there may be a Wumpus in either $Rm_{1,3}$ or $Rm_{2,2}$: $W_{1,3} \vee W_{2,2}$

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}$ **KB**

If there was a Wumpus in $Rm_{2,2}$, then **A** would have sensed a stench in $Rm_{2,1}$, but it didn't. So there is no Wumpus in $Rm_{2,2}$: $\neg W_{2,2}$

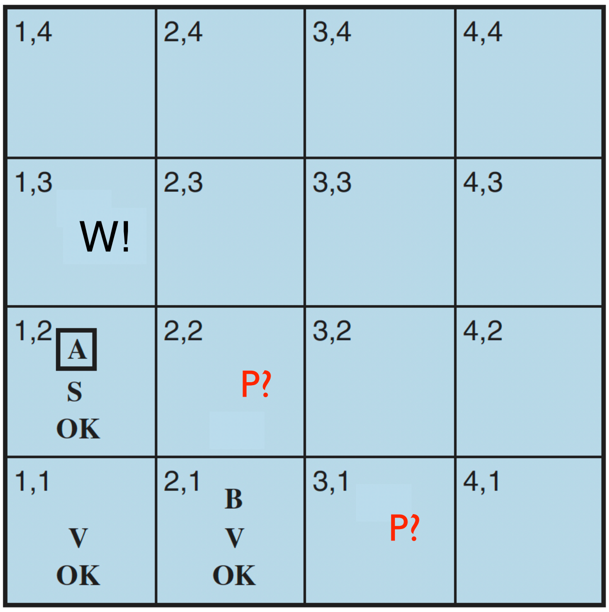
$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$ **KB**

16

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

$$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$$
$$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$$

KB



But $W_{1,3} \vee W_{2,2}$ and $\neg W_{2,2}$ imply $W_{1,3}$, so there is a Wumpus in $R_{1,3}$

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

Resolution

$$\frac{\neg W_{2,2} \qquad \neg W_{2,2} \rightarrow W_{1,3}}{W_{1,3}}$$

Modus Ponens

17

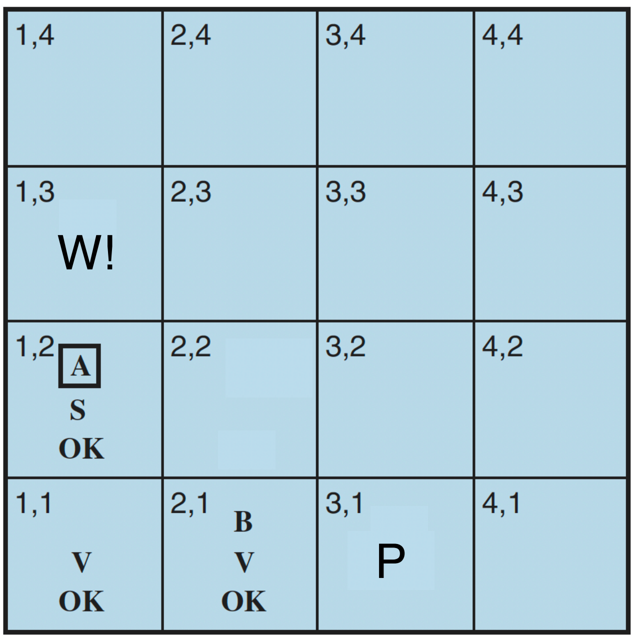
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

$$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$$
$$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$$

KB

If there was a Pit in $Rm_{2,2}$, then **A** would have sensed a breeze in $Rm_{1,2}$, but it didn't. So there is no Pit in $Rm_{2,2}$: $\neg P_{2,2}$

$$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$$
$$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$$
$$\neg P_{2,2}$$



But $P_{2,2} \vee P_{3,1}$ and $\neg P_{2,2}$ imply $P_{3,1}$, so there is a Pit in $R_{3,1}$

$$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$$
$$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$$
$$\neg P_{2,2}, P_{3,1}$$

18

A	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*

Since there is no pit and no Wumpus, $\neg P_{2,2} \wedge \neg W_{2,2}$, in $Rm_{2,2}$, it is ok: $OK_{2,2}$

KB

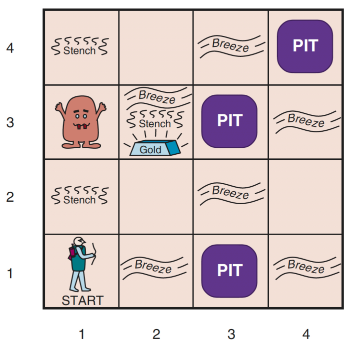
$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$
$\neg P_{2,2}, P_{3,1}, OK_{2,2}$

1,4	2,4	3,4	4,4
1,3 W!	2,3 OK	3,3	4,3
1,2 A S OK	2,2 OK	3,2 OK	4,2
1,1 V OK	2,1 B V OK	3,1 P	4,1

A chooses to move to $Rm_{2,2}$. Since there is no stench or breeze in $Rm_{2,2}$, Both $Rm_{2,3}$ and $Rm_{3,2}$ are ok to move to: $OK_{2,3} \wedge OK_{3,2}$

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$
$\neg P_{2,2}, P_{3,1}, OK_{2,2}, OK_{2,3}, OK_{3,2}$

A	= *Agent*
B	= *Breeze*
G	= *Glitter, Gold*
OK	= *Safe square*
P	= *Pit*
S	= *Stench*
V	= *Visited*
W	= *Wumpus*



KB

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$
$\neg P_{2,2}, P_{3,1}, OK_{2,2}, OK_{2,3}, OK_{3,2}$

A chooses to move to $Rm_{2,3}$ and senses a breeze, stench, and gold:

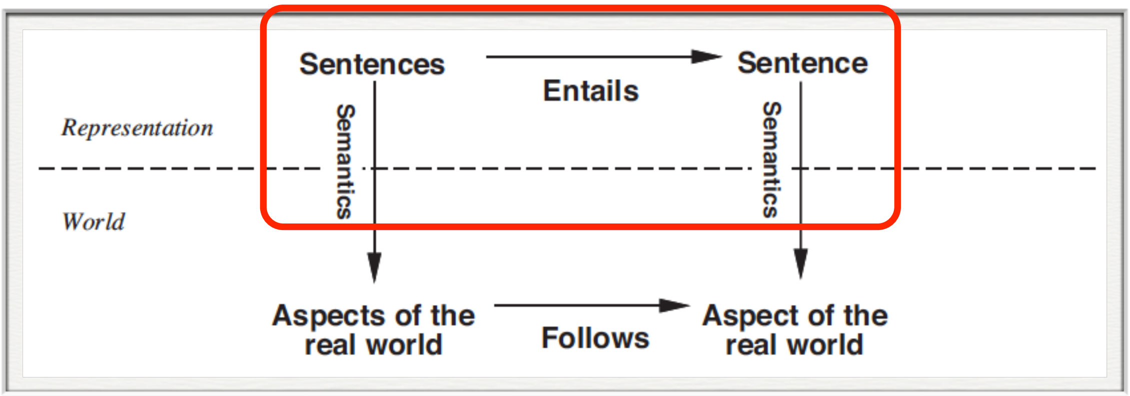
1,4	2,4 P?	3,4	4,4
1,3 W!	2,3 A S G B	3,3 P?	4,3
1,2 S V OK	2,2 V OK	3,2 OK	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

KB

$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{2,1}, OK_{1,2}$
$B_{2,1}, P_{2,2} \vee P_{3,1}, S_{1,2}, W_{1,3} \vee W_{2,2}, \neg W_{2,2}$
$\neg P_{2,2}, P_{3,1}, OK_{2,2}, OK_{2,3}, OK_{3,2}$
$B_{2,3}, S_{2,3}, G_{2,3}$

A picks up the gold, generates a motion plan to get back to [1,1] and wins the game!

Logic as a Representation Language



Propositional Logic

DPLL - Model Checking

Sat Solving

Resolution Theorem Proving

First-Order Logic

Restricted First-Order Logic

Resolution Theorem Proving

Default/ Nonmonotonic Reasoning

Answer Set Programming

Propositional Logic

Propositional Logic

The elements of the language:
Atoms: Two distinguished atoms T and F and the countably infinite set of those strings of characters that begin with a capital letter, for example, P, Q, R, . . ., P1, Q1, ON_A_B, etc.

Connectives: $\wedge$, $\vee$, $\rightarrow$, and, $\neg$, called "and", "or", "implies", and "not".

Syntax of well-formed formulas (wffs), also called ***sentences***:

- Any atom is a wff
- if $\omega 1$, $\omega 2$ are wffs, so are
- $\omega 1 \wedge \omega 2$ (conjunction)
- $\omega 1 \vee \omega 2$ (disjunction)
- $\omega 1 \rightarrow \omega 2$ (implication)
- $\neg \omega 1$ (negation)

Parentheses will be used extra-linguistically grouping wffs into sub wffs according to recursive defs

Semantics

What do sentences *mean*?

Atoms

$\downarrow$

Propositions

Semantics is about associating elements of a logical language with elements of a domain of discourse.

In the case of propositional logic, the domain of discourse is propositions about the world.

One associates atoms in the language with propositions.

An interpretation associates an atomic proposition with each atom and a value (True or False)

$P_{1,2}$

$\downarrow$

`There is a pit in Rm1,2`

α

$\downarrow$

P

If atom α is associated with proposition P, then we say that α has value *True* just in case P is true of the world; otherwise it has value *False*

The Truth Table Method

Truth tables can be used to compute the truth value of any wff given the truth values of the constituent atoms in the formula.

ω1	ω2	ω1 ∧ ω2	ω1 ∨ ω2	¬ω1	ω1→ω2
True	True	True	True	False	True
True	False	False	True	False	False
False	True	False	True	True	True
False	False	False	False	True	True

$[(P \rightarrow Q) \rightarrow R] \rightarrow P$

P is False
Q is False
R is True

Interpretation

If an agent describes its world using n features (corresponding to propositions) and these features are represented as n atoms in the agent's model of the world then there are 2^n ways the world can be as far as the agent can discern/express.

LINKÖPING UNIVERSITY

Satisfiability and Models

An interpretation satisfies a *wff* if the *wff* is assigned the value True under the interpretation.

An interpretation that satisfies a *wff* (set of *wff*s) is called a model of the *wff* (set of *wff*s).

Find an interpretation that is a model of: $P_{1,2} \vee W_{1,2} \rightarrow \neg OK_{1,2}$

A wff is said to be inconsistent or unsatisfiable if there are no interpretations that satisfy it. (Likewise for sets of sentences)

$P_{1,2} \wedge \neg P_{1,2}$ $\{P_{1,2} \vee W_{1,2}, P_{1,2} \vee \neg W_{1,2}, \neg P_{1,2} \vee W_{1,2}, \neg P_{1,2} \vee \neg W_{1,2}\}$

LINKÖPING UNIVERSITY

Validity and Entailment

A wff is said to be valid if it has value *True* under all interpretations of its constituent atoms.

Are the following valid sentences? $\neg(P_{1,2} \wedge \neg P_{1,2})$ $\neg(P_{1,2} \wedge \neg W_{1,2})$

If a wff ω has value *True* under all those interpretations for which each of the wffs in a set Δ has value *True*, then we say that Δ logically entails ω and that ω logically follows from Δ and that ω is a logical consequence of Δ. We use the symbol $\models$ to denote logical entailment and write $\Delta \models \omega$

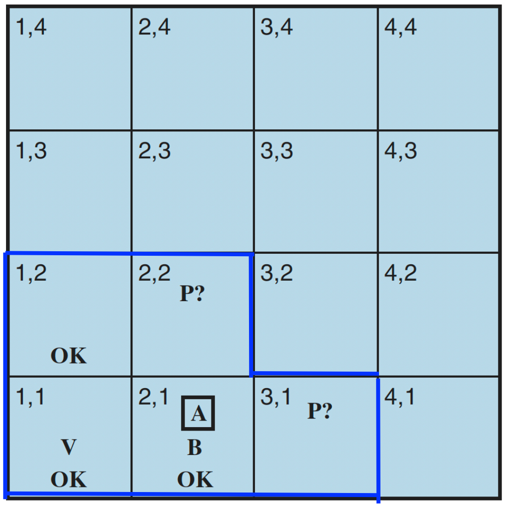
$\{P_{1,2}\} \models P_{1,2}$ $\{\} \models \neg(P_{1,2} \wedge \neg P_{1,2})$

$\{P_{1,2}, P_{1,2} \rightarrow W_{1,2}\} \models W_{1,2}$

False $\models \omega$ where ω is any wff!

Require an efficient means of testing whether sentences are True in an interpretation and whether sentences are entailed by sets of sentences.

An Entailment Example



Lets restrict ourselves to the blue cells: [1,1], [2,1], [3,1],[1,2],[2,2]

We want to reason about PITS in: [1,2],[2,2],[3,1]

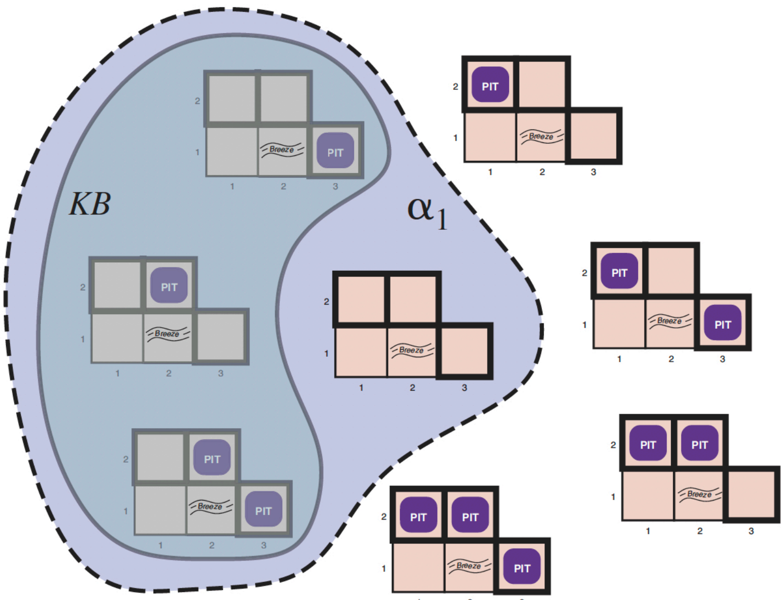
There are 8 possibilities: pit or no pit. Consequently, 8 possible models for the presence/non-presence of pits

But our percepts together with the rules of the game restrict us to three possible models satisfying the KB

$$\neg B_{1,1}, \neg S_{1,1}, OK_{1,1}, OK_{1,2}, OK_{2,1}, B_{2,1}, P_{2,2} \vee P_{3,1}$$

Rules of The game: $OK_{x,y} \rightarrow \neg P_{x,y}$ .... $OK_{x,y} \rightarrow \neg S_{x,y}$

Wumpus Possible Worlds



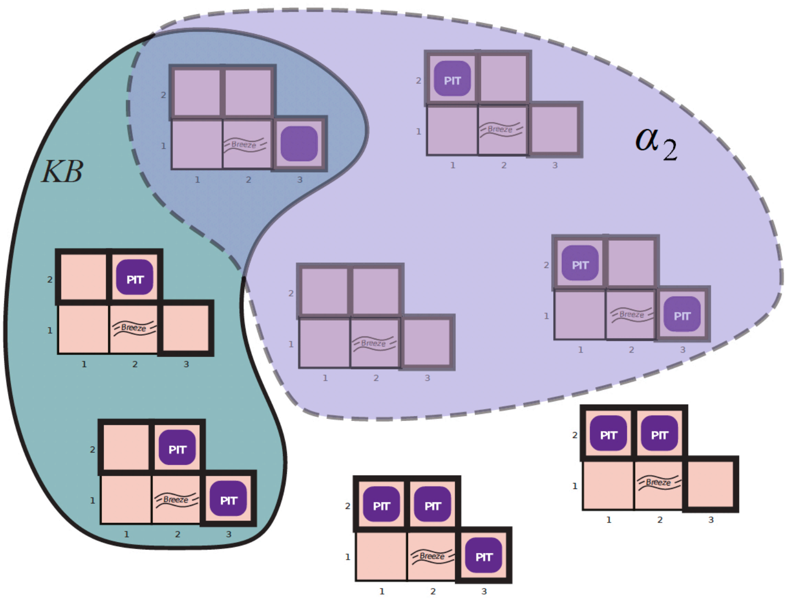
Suppose:

$$\alpha_1 = \neg P_{1,2}$$

Is α_1 entailed by KB?

YES! $KB \models \alpha_1$

Wumpus Possible Worlds



Suppose:

$$\alpha_2 = \neg P_{2,2}$$

Is α_2 entailed by KB?

NO! $KB \models \alpha_2$

Truth Table Enumeration

- Enumerate all models
- Check that the query is is true in all models that satisfy the KB

Entailment checking by enumeration

Model checking approach

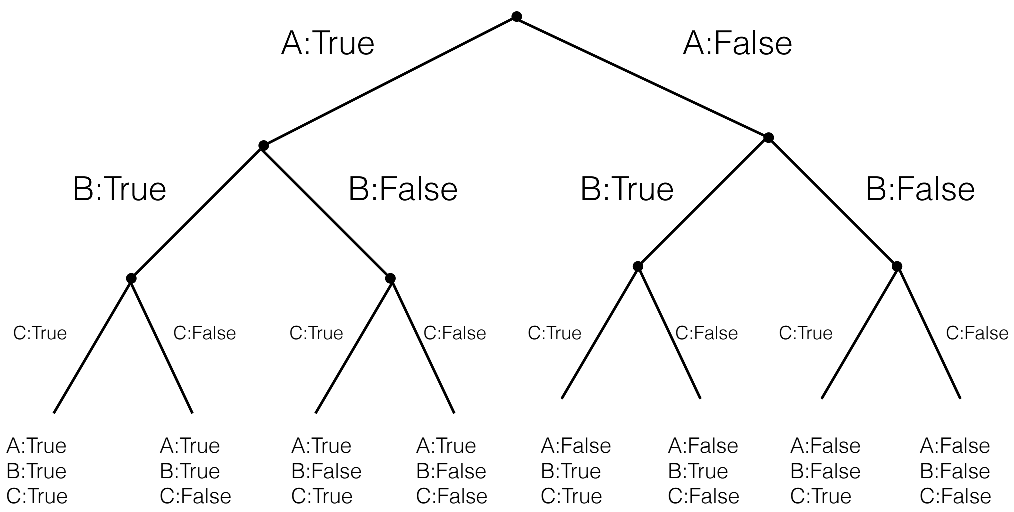
- Recursively build tree where each leaf is a model.
- Check that:
  - Each model that makes KB true, makes query true.

```
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        //  when KB is false, always return true
   else
      P ← FIRST(symbols)
      rest ← REST(symbols)
      return (TT-CHECK-ALL(KB, α, rest, model ∪ {P = true})
              and
              TT-CHECK-ALL(KB, α, rest, model ∪ {P = false }))
```

LINKÖPING UNIVERSITY



Proof Theory

Straightforward model-checking approaches are generally not efficient since the number of models grows exponentially with the number of variables.

Can we find a more efficient "syntactic" means of of showing semantic consequence without the need to generate models?

We also have to "guarantee" that the syntactic approach is equivalent to the semantic approach.

For when I am presented with a false theorem, I do not need to examine or even to know the demonstration, since I shall discover its falsity a posteriori by means of an easy experiment, that is by calculation, costing no more than paper and ink, which will show the error no matter how small it is…

And if someone would doubt my results, I should say to him:

"Let us calculate, Sir", and thus by taking paper to pen and ink, we should soon settle the question

–Gottfried Wilhelm Leibniz [1677]

Calculus Ratiocinator

Rules of Inference and Proofs

Now that we have a feeling for the intuitions behind entailment and its potential, the next step is to find syntactic characterizations of the reasoning process (inference) to make this functionality feasible for use in intelligent agents. We require a **proof theory**.

Rules of inference permit us to produce additional wffs from others in a **sound** or **truth-preserving** manner.

If what comes in is true, then what comes out is true

Some Examples:

$$\frac{\omega_1, \omega_2}{\omega_1 \wedge \omega_2} \qquad \frac{\omega_1, \omega_1 \rightarrow \omega_2}{\omega_2}$$

LINKÖPING UNIVERSITY

Definition of a Proof

The sequence of wffs $\{\omega_1, \omega_2, \ldots, \omega_n\}$ is called a proof (or deduction) of ω_n from a set of wffs Δ *iff* each ω_i in the sequence is either

- in Δ, or
- can be inferred from a *wff* (or *wffs*) earlier in the sequence by using one of the rules of inference (in the proof theory).

Proof of $Q \wedge R$ from Δ

$\Delta = \{P, R, P \rightarrow Q\}$

$\{P, P \rightarrow Q, Q, R, Q \wedge R\}$

If there is a proof of ω_n from Δ, we say that ω_n is a theorem of the set Δ. The following notation will be used for expressing that ω_n can be proved from Δ: $\Delta \vdash \omega_n$

(or $\Delta \vdash_{\mathscr{R}} \omega_n$, where $\mathscr{R}$ refers to a set of inference rules

$\Delta \vdash Q \wedge R$

Natural Deduction

LINKÖPING UNIVERSITY

Soundness and Completeness

If, for any set of wffs, Δ , and wff, ω, $\Delta \vdash_{\mathfrak{R}} \omega$ implies $\Delta \models \omega$, we say that the set of inference rules, $\mathfrak{R}$, is **sound**.

If, for any set of wffs, Δ , and wff, ω, it is the case that whenever $\Delta \models \omega$, there exists a proof of ω from Δ using the set of inference rules, $\mathfrak{R}$, we say that $\mathfrak{R}$ is **complete**.

Syntactic characterizations of Entailment

Soundness -- not too strong!
Completeness -- not too weak!

LINKÖPING UNIVERSITY

37

Some Important Meta-Theorems

The Deduction Theorem

if $\{\omega 1, \omega 2, \ldots, \omega n\} \models \omega$ then $(\omega 1 \wedge \omega 2 \wedge \ldots \omega n) \rightarrow \omega$ is valid and vice-versa.

Can transform a question of entailment into a question of validity

Reductio ad absurdum

If the set Δ has a model but $\Delta \cup \{\neg\omega\}$ does not, then $\Delta \models \omega$

Proof by Refutation: To prove that $\Delta \models \omega$, show that $\Delta \cup \{\neg\omega\}$ has no model. [Unsatisfiable]

Can transform a question of entailment into a question of satisfiability!

LINKÖPING UNIVERSITY

38

Efficient Propositional Model Checking

DPLL

LINKÖPING UNIVERSITY

Clauses and Normal Forms

A literal is an atom (positive literal) or the negation of an atom (negative literal)

$B_{2,3}, \neg P_{3,4}$

A clause is an expression of the form:

$$l_1 \vee l_2 \vee \ldots \vee l_k$$

$P_{3,1} \vee \neg W_{2,2} \vee B_{1,3}$

where each l_i is a literal

A wff written as a <u>*conjunction*</u> *of clauses* is said to be in conjunctive normal form (**CNF**).

$(P_{3,1} \vee \neg W_{2,2} \vee B_{1,3}) \wedge (\neg B_{2,3} \vee W_{3,3}) \wedge S_{2,2}$

A wff written as a <u>*disjunction*</u> *of conjunctions of literals* is said to be in disjunctive normal form (**DNF**).

Any propositional formula can be converted into an equivalent CNF or DNF form

LINKÖPING UNIVERSITY

Converting to CNF or DNF form

1. Eliminate implication connectives by using the equivalent form with ¬,V.
2. Reduce the scope of ¬ connectives by applying DeMorgan's laws and by eliminating double negations (¬¬) if they arise.
3. Convert to CNF(DNF) by using associative and distributive laws.

$$\neg(\omega_1 \vee \omega_2) \equiv \neg\omega_1 \wedge \neg\omega_2$$
$$\neg(\omega_1 \wedge \omega_2) \equiv \neg\omega_1 \vee \neg\omega_2$$

DeMorgan Laws

$$\omega_1 \wedge (\omega_2 \vee \omega_3) \equiv (\omega_1 \wedge \omega_2) \vee (\omega_1 \wedge \omega_3)$$
$$\omega_1 \vee (\omega_2 \wedge \omega_3) \equiv (\omega_1 \vee \omega_2) \wedge (\omega_1 \vee \omega_3)$$

Distributive Laws

$$(\omega_1 \wedge \omega_2) \wedge \omega_3 \equiv \omega_1 \wedge (\omega_2 \wedge \omega_3)$$
$$(\omega_1 \vee \omega_2) \vee \omega_3 \equiv \omega_1 \vee (\omega_2 \vee \omega_3)$$

Associative Laws

LINKÖPING UNIVERSITY

An Example

$$\neg(P \rightarrow Q) \vee (R \rightarrow P)$$

$\neg(\neg P \vee Q) \vee (\neg R \vee P)$ Eliminate implication connectives

$(P \wedge \neg Q) \vee (\neg R \vee P)$ Apply DeMorgan's Law

$(P \vee \neg R \vee P) \wedge (\neg Q \vee \neg R \vee P)$ Apply Distributive Law

$(P \vee \neg R) \wedge (\neg Q \vee \neg R \vee P)$ Factor (remove duplicates)

$$\neg(P \rightarrow Q) \vee (R \rightarrow P) \equiv (P \vee \neg R) \wedge (\neg Q \vee \neg R \vee P)$$

CNF Form

A conjunction of clauses

LINKÖPING UNIVERSITY

Davis Putnam Algorithm

- The Davis-Putnam Algorithm (1960)
  - In a seminal paper, they described an effective satisfiability checking algorithm
  - Satisfiability by search
  - Takes as input a formula in conjunctive normal form (set of clauses)
- The Davis, Putnam, Logeman, Loveland Algorithm (1962) DPLL
  - An extension of the DP algorithm with better space efficiency
- Essentially a recursive, depth-first enumeration of possible models with three improvements over TT-ENTAILS
  - *Early Termination*
  - *Pure Symbol Heuristic*
  - *Unit Clause Heuristic*
- Most modern SAT solvers are still based on ideas from DPLL

LINKÖPING UNIVERSITY

Some notation

A partial assignment is a mapping from a set of variables to truth values : $\varphi : V \rightarrow \{\text{true}, \text{false}\}$

An application of a partial assignment to a clause set *F* is denoted by: $\varphi * F$

It results in the clause set obtained from *F* by first removing all clauses satisfied by φ , and then removing from the remaining clauses all literal occurrences which are falsified by φ

$\varphi : \{\text{A : true, D : false}\}$

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

$$(\text{true} \vee \neg B) \wedge (\neg B \vee \neg C) \wedge (C \vee \text{false})$$

$$(\neg B \vee \neg C) \wedge (C \vee \text{false})$$

$$(\neg B \vee \neg C) \wedge (C)$$

$$\{\{A, \neg B\}, \{\neg B, C\}, \{C, D\}\}$$

$$\{\{\neg B, C\}, \{C\}\}$$

A partial assignment φ is a weak autarchy for *F* if: $\varphi * F \subseteq F$

If φ is a weak autarchy for F, then $\varphi * F$ is satisfiability equivalent to F

If I can satisfy the remaining clauses in $\varphi * F$ then F is satisfiable too

LINKÖPING UNIVERSITY

Early Termination

If A is true in an assignment then

$\varphi : \{A : \text{true}\}$ $$(A \vee B \vee D) \wedge (A \vee \neg E \vee F) \wedge (A \vee G)$$

$$(\text{true} \vee B \vee D) \wedge (\text{true} \vee \neg E \vee F) \wedge (\text{true} \vee G)$$

is true without knowing the assignment of other variables.

If A and G are false in an assignment then

$\varphi : \{A : \text{false}, G : \text{false}\}$ $$(A \vee B \vee D) \wedge (A \vee \neg E \vee F) \wedge (A \vee G)$$

$$(A \vee B \vee D) \wedge (A \vee \neg E \vee F) \wedge (\text{false} \vee \text{false})$$

$$(A \vee B \vee D) \wedge (A \vee \neg E \vee F) \wedge (\text{false})$$

is false without knowing the assignment of other variables.

LINKÖPING UNIVERSITY

Pure Symbol Heuristic

A “pure” symbol is a symbol that always appears with the same sign in all clauses

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

A is pure, B is pure and C is not

Assigning a pure symbol the value that makes it true will never make the original clause false

$\varphi : \{A : \text{true}\}$ $$(\text{True} \vee \neg B) \wedge (\neg B \vee \neg C) \wedge (C \vee \text{True})$$

$$(\neg B \vee \neg C)$$

φ is a weak autarchy for F : $\varphi * F$ is satisfiability equivalent to F

φ is a weak autarchy for F : $\varphi * F$ is unsatisfiability equivalent to F

LINKÖPING UNIVERSITY

Unit Clause Heuristic

Unit clause in resolution: A clause with one literal

Unit clause in DPLL: also means clauses in which all literals but one are already assigned false by the model

$\varphi : \{A : \text{true}, B : \text{false}\}$ $\quad (\neg A \vee B \vee C) \wedge (D \vee E) \wedge (\neg C \vee F) \wedge G$

$(\text{False} \vee \text{False} \vee C) \wedge (D \vee E) \wedge (\neg C \vee F) \wedge G$

For a unit clause to be true, it must have one assignment.

Unit Clause Heuristic: Assign all such symbols before branching on the remainder

LINKÖPING UNIVERSITY

Example

$\varphi : \{A : \text{true}, B : \text{false}\}$ $\quad (\neg A \vee B \vee C) \wedge (D \vee E) \wedge (\neg C \vee \neg F) \wedge G$

$(\text{false} \vee \text{false} \vee C) \wedge (D \vee E) \wedge (\neg C \vee \neg F) \wedge G$

$C \wedge (D \vee E) \wedge (\neg C \vee \neg F) \wedge G$

$\varphi : \{A : \text{true}, B : \text{false}, G : \text{true}\}$

$C \wedge (D \vee E) \wedge (\neg C \vee \neg F) \wedge \text{true}$

$C \wedge (D \vee E) \wedge (\neg C \vee \neg F)$

$\varphi : \{A : \text{true}, B : \text{false}, G : \text{true}, C : \text{true}\}$

$C \wedge (D \vee E) \wedge (\neg C \vee \neg F)$

$\text{true} \wedge (D \vee E) \wedge (\text{false} \vee \neg F)$

$(D \vee E) \wedge \neg F$

$\varphi : \{A : \text{true}, B : \text{false}, G : \text{true}, C : \text{true}, F : \text{false}\}$

$(D \vee E) \wedge \text{true}$

$(D \vee E)$

LINKÖPING UNIVERSITY

The DPLL Algorithm

function DPLL-SATISFIABLE?(s) **returns** *true* or *false*
  inputs: s, a sentence in propositional logic

  clauses ← the set of clauses in the CNF representation of s
  symbols ← a list of the proposition symbols in s
  return DPLL(*clauses*, *symbols*, { })

function DPLL(*clauses*, *symbols*, *model*) **returns** *true* or *false*

  if every clause in *clauses* is true in *model* **then return** *true*
  if some clause in *clauses* is false in *model* **then return** *false*
  P, *value* ← FIND-PURE-SYMBOL(*symbols*, *clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P, *value* ← FIND-UNIT-CLAUSE(*clauses*, *model*)
  if P is non-null **then return** DPLL(*clauses*, *symbols* – P, *model* ∪ {P=*value*})
  P ← FIRST(*symbols*); *rest* ← REST(*symbols*)
  return DPLL(*clauses*, *rest*, *model* ∪ {P=*true*}) **or**
      DPLL(*clauses*, *rest*, *model* ∪ {P=*false*})

Detects early termination for partially completed models

Splitting Rule

Provides a skeleton of the search process.

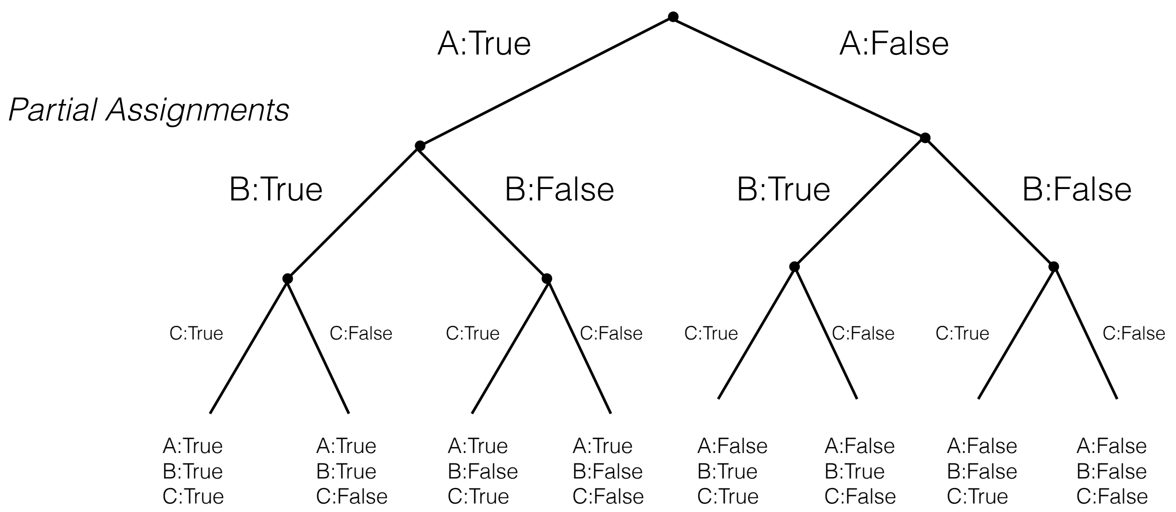
Note1: Each application of a heuristic includes simplifying the clause set

Note2: Each application of a heuristic is satisfiability preserving

LINKÖPING UNIVERSITY

DPLL is similar to TT-Entails

Recursive depth-first search



Uses heuristics so the whole tree may not need be expanded and searched. Stops when it finds a solution

Using DPLL for Inference

Want to know whether: $\Delta \vDash \alpha$

Want to turn this into a satisfiability problem!

Deduction Theorem: If $\Delta \vDash \alpha$ then $\vDash \Delta \rightarrow \alpha$

$\Delta \rightarrow \alpha$ is valid iff $\neg(\Delta \rightarrow \alpha)$ $(= \Delta \wedge \neg\alpha)$ is unsatisfiable

Let β be $\Delta \wedge \neg\alpha$ in CNF form

If DPPL-Satisfiable?(β) is true then $\Delta \vDash \alpha$ is false

If DPPL-Satisfiable?(β) is false then $\Delta \vDash \alpha$ is true

LINKÖPING UNIVERSITY

Recent Extensions to DPLL

- *Component Analysis*
  - Find independent subsets of unassigned variables (components) and solve each component separately
- *Variable and Value Ordering*
  - degree heuristic - choose a variable appearing most frequently among remaining clauses
  - choose true or false as an assignment heuristically
- *Intelligent backtracking*
  - Also do conflict clause learning
- *Random restarts*
  - If little progress in extending an assignment, random restart
    - remember clauses assigned, change variable and value selection
- *Clever indexing techniques*
  - acquiring clause types rapidly…

LINKÖPING UNIVERSITY

Axiomatizing the Wumpus World

Physics of the Wumpus World:
Modeling is difficult with Propositional Logic

Schemas:

$(B_{x,y} \Leftrightarrow (P_{x,y+1} \lor P_{x,y-1} \lor P_{x+1,y} \lor P_{x-1,y}))$ Def. of breeze in pos [x,y]

$(S_{x,y} \Leftrightarrow (W_{x,y+1} \lor W_{x,y-1} \lor W_{x+1,y} \lor W_{x-1,y}))$ Def. of stench in pos [x,y]

$(W_{1,1} \lor W_{1,2} \lor \ldots \lor W_{4,4}))$ There is at least one wumpus!

..., etc. There is only one wumpus!

LINKÖPING UNIVERSITY

Logical Wumpus Hybrid Agent

function HYBRID-WUMPUS-AGENT(*percept*) **returns** an *action*
  inputs: *percept*, a list, [*stench*,*breeze*,*glitter*,*bump*,*scream*]
  persistent: *KB*, a knowledge base, initially the atemporal "wumpus physics"
    t, a counter, initially 0, indicating time
    plan, an action sequence, initially empty

  TELL(*KB*, MAKE-PERCEPT-SENTENCE(*percept*, *t*))
  TELL the *KB* the temporal "physics" sentences for time *t*
  $safe \leftarrow \{[x, y] \ : \ \text{ASK}(KB, OK^t_{x,y}) = true\}$
  if $\text{ASK}(KB, Glitter^t) = true$ **then**
    $plan \leftarrow [Grab] + \text{PLAN-ROUTE}(current, \{[1,1]\}, safe) + [Climb]$
  if *plan* is empty **then** Plan a route to an unvisited cell through safe cells
    $unvisited \leftarrow \{[x, y] \ : \ \text{ASK}(KB, L^{t'}_{x,y}) = false$ for all $t' \leq t\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap safe, safe)$
  if *plan* is empty and $\text{ASK}(KB, HaveArrow^t) = true$ **then** Plan a route to To make safe cells
    $possible\_wumpus \leftarrow \{[x, y] \ : \ \text{ASK}(KB, \neg W_{x,y}) = false\}$
    $plan \leftarrow \text{PLAN-SHOT}(current, possible\_wumpus, safe)$
  if *plan* is empty **then** // *no choice but to take a risk* Plan a route to an unvisited cell through not unsafe cells
    $not\_unsafe \leftarrow \{[x, y] \ : \ \text{ASK}(KB, \neg OK^t_{x,y}) = false\}$
    $plan \leftarrow \text{PLAN-ROUTE}(current, unvisited \cap not\_unsafe, safe)$
  if *plan* is empty **then** Give up: Plan a route to start cell through safe cells
    $plan \leftarrow \text{PLAN-ROUTE}(current, \{[1, 1]\}, safe) + [Climb]$
  $action \leftarrow \text{POP}(plan)$
  TELL(*KB*, MAKE-ACTION-SENTENCE(*action*, *t*))
  $t \leftarrow t + 1$
  return *action*

$L^t_{x,y}$: Visited [x,y] at time t

Successor state axioms, etc

Try to construct plans based on goals with decreasing priority

function PLAN-ROUTE(*current*,*goals*,*allowed*) **returns** an action sequence
  inputs: *current*, the agent's current position
    goals, a set of squares; try to plan a route to one of them
    allowed, a set of squares that can form part of the route

  $problem \leftarrow \text{ROUTE-PROBLEM}(current, goals, allowed)$
  return SEARCH(*problem*) // *Any search algorithm from Chapter ??*

$\text{ASK}(KB, \neg W_{x,y}) = false$

means that $KB \models \neg W_{x,y}$ is false

It does not mean that $KB \models W_{x,y}$ is true

$\text{ASK}(KB, \neg OK^t_{x,y}) = false$

means that $KB \models \neg OK^t_{x,y}$ is false

It does not mean that $KB \models OK^t_{x,y}$ is true

LINKÖPING UNIVERSITY

Local Search Algorithms for SAT

- Studied local search algorithms previously that combine both greediness and randomness
  - Hill climbing
  - Simulated Annealing
  - Stochastic Beam Search
- Local search can be applied directly to the SAT problem
  - Find an assignment that satisfies all clauses
  - Instances (states) are full assignments
  - Children generated by flipping a variable's assignment (T to F or F to T)
  - Evaluation function -
    - count the number of unsatisfied clauses (in the CNF)
    - minimize that number
  - Can be many local minima
    - Use randomness to escape

LINKÖPING UNIVERSITY

WalkSAT

On each iteration: pick an unsatisfied clause and pick a symbol in it to flip

function WALKSAT(*clauses*, *p*, *max_flips*) **returns** a satisfying model or *failure*
  inputs: *clauses*, a set of clauses in propositional logic
    p, the probability of choosing to do a "random walk" move, typically around 0.5
    max_flips, number of value flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for each $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    if RANDOM$(0, 1) \le p$ **then**
      flip the value in *model* of a randomly selected symbol from *clause*
    else flip whichever symbol in *clause* maximizes the number of satisfied clauses
  return *failure*

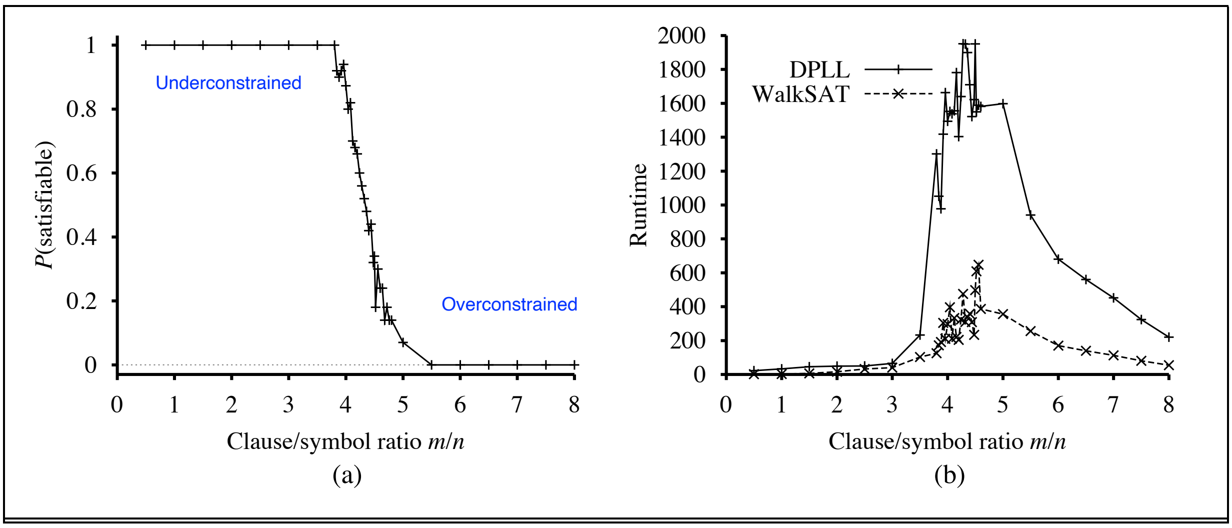
Two ways to choose the symbol to flip:
- *min-conflicts*: minimize the number of unsatisfied clauses
- *random walk*: pick the symbol randomly

LINKÖPING UNIVERSITY

WalkSAT: Completeness and Termination

- **WalkSAT** is sound
  - When the algorithm returns a model it does satisfy the input clauses.
- **WalkSAT** is not complete
  - When **WalkSAT** fails
    - either the sentence is unsatisfiable, or
    - the algorithm needs more time to find a solution
  - *max_flips* = infinity and p > 0
    - if a model exists, it will eventually find it (random walk)
    - if a model does not exist, the algorithm never terminates.
- SAT is **NP-Complete,** so some problem instances will require exponential runtime.
  - Can we delineate the hard problem instances from the easy problem instances?

Landscape of Random SAT Problems



(a) Graph showing the probability that a random 3-CNF sentence with n = 50 symbols is satisfiable, as a function of the clause/symbol ratio m/n.

(b) Graph of the median run time (measured in number of recursive calls to DPLL, a good proxy) on random 3-CNF sentences.

The most difficult problems have a clause/symbol ratio of about 4.3.

$CNF_3(m,50)$

Sentences with 50 variables and 3 literals per clause

5 symbols/5clauses:

$$(\neg D \lor \neg B \lor C) \land (B \lor \neg A \lor \neg C) \land (\neg C \lor \neg B \lor E) \land (E \lor \neg D \lor B) \land (\neg B \lor E \lor \neg C)$$

Resolution Theorem Proving

Resolution Theorem Proving

We considered the unit clause heuristic [1960 Davis Putnam] when studying DPLL and found that it is a satisfiability/truth preserving heuristic.

Robinson [1965], in a major breakthrough in automated theorem proving, based his technique on the resolution inference rule and also generalised it for the 1st-order case by introducing "on-demand" grounding using a unification algorithm.

Let's begin with Unit Resolution

A unit clause is a disjunction/clause consisting of a single literal

The unit resolution rule takes a clause and a unit clause and returns a new clause called the resolvant.

Resolution Rules

Unit Resolution Rule:

$$\frac{l_1 \vee \ldots \vee l_k, \quad m}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots \vee l_k}$$

The rule resolves on complementary literals: l_i, m

Example:

$$\frac{A \vee B \vee C \vee D \qquad \neg C}{A \vee B \vee D}$$

Example:

$$\frac{A \vee B \vee C \vee D \qquad \neg B}{A \vee C \vee D}$$

LINKÖPING UNIVERSITY

Resolution Rules

The unit resolution rule can be generalised:

$$\frac{l_1 \vee \ldots \vee l_k, \quad m_1 \vee \ldots \vee m_n}{l_1 \vee \ldots \vee l_{i-1} \vee l_{i+1} \vee \ldots l_k \vee m_1 \vee \ldots \vee m_{j-1} \vee m_{j+1} \vee \ldots \vee m_n}$$

where l_i and m_j are complementary literals

Example:

$$\frac{A \vee B \vee C \vee D \qquad \neg E \vee \neg C \vee F}{A \vee B \vee D \vee \neg E \vee F}$$

LINKÖPING UNIVERSITY

Resolution

Note: The empty clause is **False**

An empty disjunction is false by definition
We sometimes use {} also

$$\frac{P, \quad \neg P}{\bot}$$

Forward Rule chaining is a special case of Resolution:

If R → P and P → Q then R → Q

$$\frac{(\neg R \lor P) \qquad (\neg P \lor Q)}{(\neg R \lor Q)}$$

Soundness of Resolution

The Resolution Rule is Sound:

$$\frac{l_1 \lor \ldots \lor l_k, \quad m_1 \lor \ldots \lor m_n}{l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots l_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n}$$

$$\{l_1 \lor \ldots \lor l_k, m_1 \lor \ldots \lor m_n\} \vdash_{\mathscr{R}}$$

$$\{l_1 \lor \ldots \lor l_{i-1} \lor l_{i+1} \lor \ldots l_k \lor m_1 \lor \ldots \lor m_{j-1} \lor m_{j+1} \lor \ldots \lor m_n\}$$

Preserves Truth and Satisfiability

Completeness of Resolution

Resolution as is, is not complete!

$$\frac{P \quad R}{P \lor R}$$?

$\{P, R\} \models P \lor R$ and it is not the case that $\{P, R\} \vdash_{\mathscr{R}} P \lor R$

Resolution can not be used
directly to decide all logical entailments….

But…..

Resolution Refutation is Complete

Recall our meta-theorem:

Reductio ad absurdum

If the set Δ has a model but $\Delta \cup \{\neg\omega\}$ does not, then $\Delta \models \omega$

Proof by Refutation: To prove that $\Delta \models \omega$, show that $\Delta \cup \{\neg\omega\}$ has no model.

We can show that the negation of $P \lor R$ $(\neg P \land \neg R)$
is inconsistent with $(P) \land (R)$

Resolve on P or R to
generate a contradiction:

$$\frac{R, \quad \neg R}{\bot} \qquad \frac{P, \quad \neg P}{\bot}$$

Resolution Refutation Procedure

To prove an arbitrary wff, ω, from a set of wffs Δ , proceed as follows:

1. Convert the wffs in Δ to clause form -- a (conjunctive) set of clauses.
2. Convert the negation of the wff to be proved, ω , to clause form.
3. Combine the clauses from steps 1 and 2 into a single set, Γ.
4. Iteratively apply resolution to the clauses in Γ and add the results to Γ either until there are no more resolvents that can be added or until the empty clause is produced.

The empty clause will be produced by the refutation resolution procedure if $\Delta \models \omega$. We say that propositional resolution is **refutation complete**.

If Δ is a finite set of clauses and if $\Delta \not\models \omega$, then the resolution refutation procedure will terminate without producing the empty clause. We say that entailment is **decidable** for the propositional calculus by resolution refutation.

A Resolution Example

Suppose the agent is in [1,1] and there is no breeze. Show that there is no pit in [1,2]

$KB = \{(B_{1,1} \leftrightarrow (P_{1,2} \vee P_{2,1}), \neg B_{1,1}\}$

Prove $\neg P_{1,2}$

Show that $KB \wedge \neg\neg P_{1,2}$ is inconsistent

$KB \wedge \neg\neg P_{1,2}$ in CNF form:

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

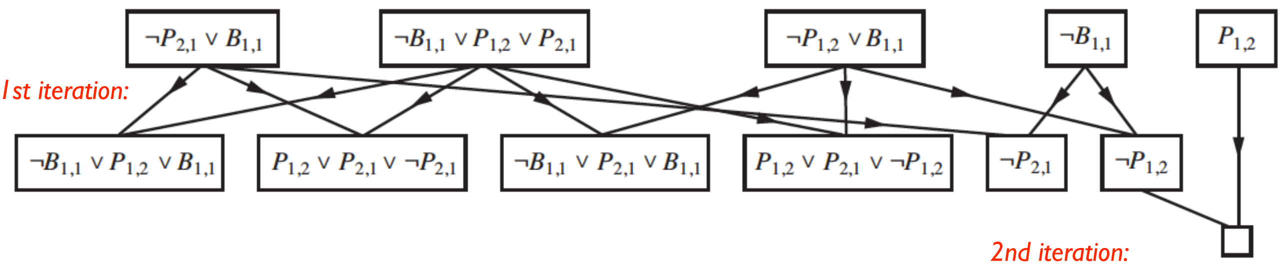
A Resolution Algorithm

```
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 ← { }
   while true do
      for each pair of clauses Ci, Cj in clauses do
         resolvents ← PL-RESOLVE(Ci, Cj)
         if resolvents contains the empty clause then return true
         new ← new ∪ resolvents
      if new ⊆ clauses then return false      No new clauses generated, no empty clause
      clauses ← clauses ∪ new
```


Applying the Algorithm

Algorithm:



$KB \wedge \neg\neg P_{1,2}$

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

Refutation Tree: