TDDC17

Seminar 5
Introduction to Knowledge Representation
Logical Agents - Wumpus World
Propositional Logic
Resolution Theorem Proving

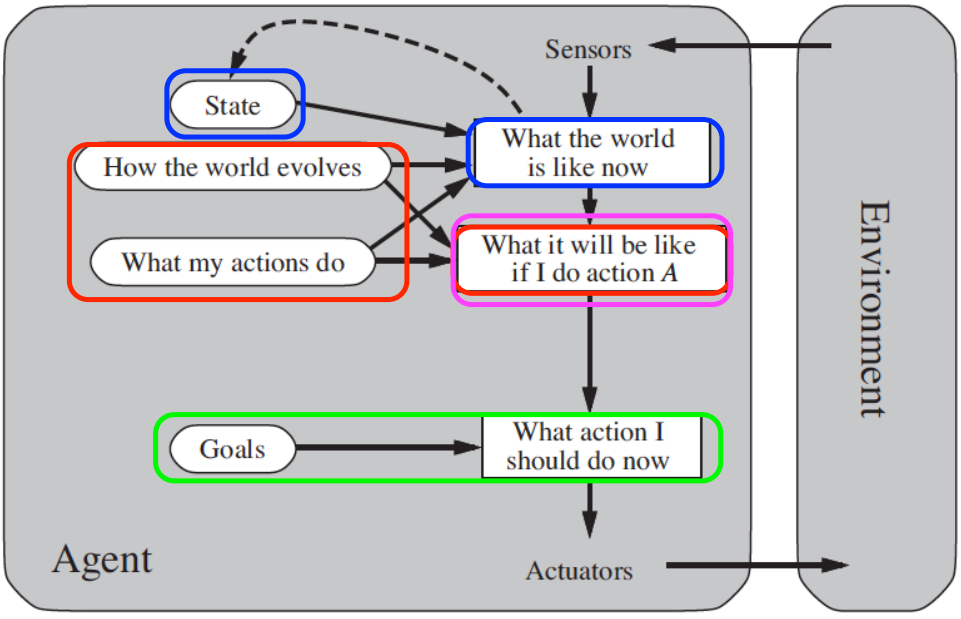
Artificial Intelligence & Integrated Computer Systems Division
Department of Computer and Information Science
Linköping University, Sweden

Model-Based, Goal-Based Agent

Keeps track of world-states

Has knowledge about the ways of the world

It is goal directed

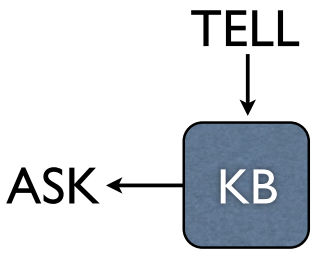


Anticipates by internal simulation/inference

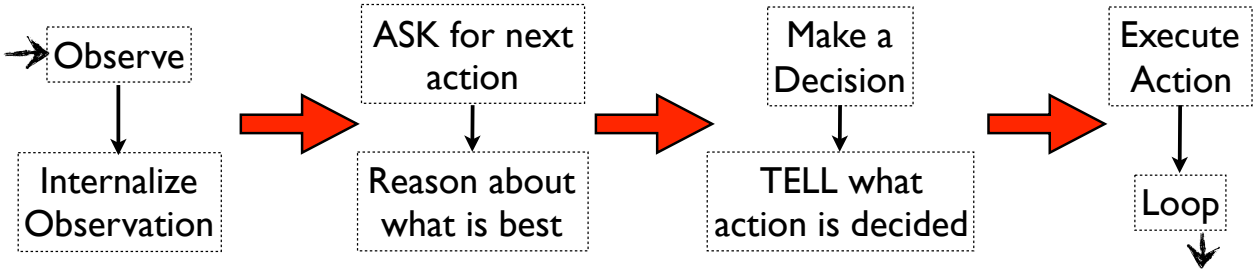
Artificial Intelligence & Integrated Computer Systems Division
Department of Computer and Information Science
Linköping University, Sweden

Generic Knowledge-Based Agent

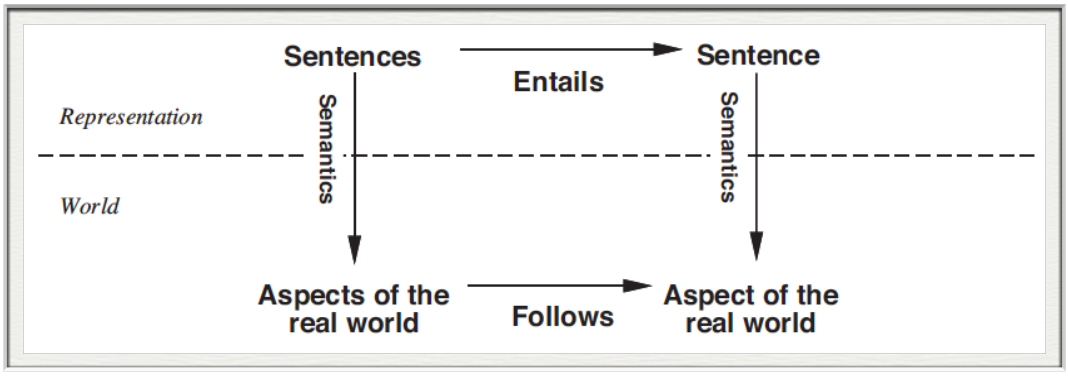
```
function KB-AGENT(percept) returns an action
  persistent: KB, a knowledge base
              t, a counter, initially 0, indicating time

  TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))
  action ← ASK(KB, MAKE-ACTION-QUERY(t))
  TELL(KB, MAKE-ACTION-SENTENCE(action, t))
  t ← t + 1
  return action
```



Advice Taker [McCarthy]



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!!

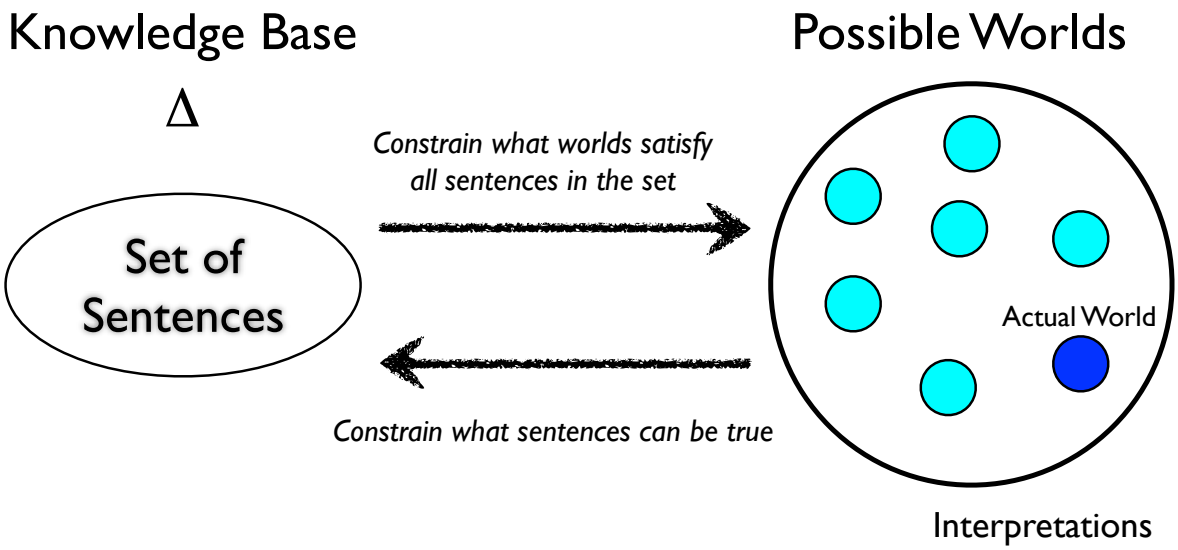
Knowledge Representation Hypothesis

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. [Smith, 1982]

Sentences as Constraints!



Syntax

Δ |--R ω

Inference

Semantics

Δ |= ω

Entailment

Logic as a Representation Language

What is Logic?

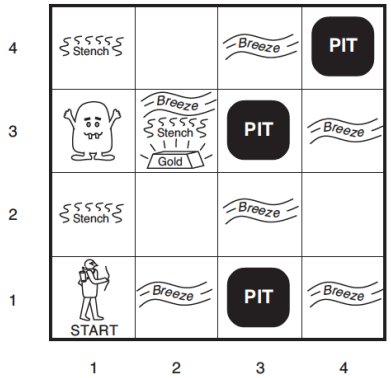
Given a set of facts Δ taken to hold as true about the ”world” and given an assertion α about the ”world”, is there a good argument for believing that α holds based on the initial set of facts Δ?

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.

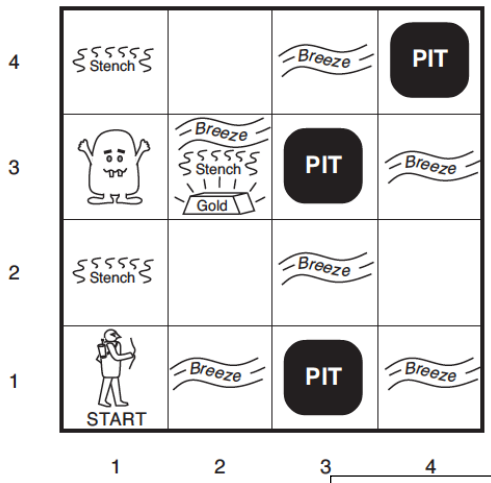
Logic in the more restricted sense is about the study of mathematical theories for formalizing the distinction between good/bad arguments and mechanizing ways to make these distinctions

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.



The Task Environment



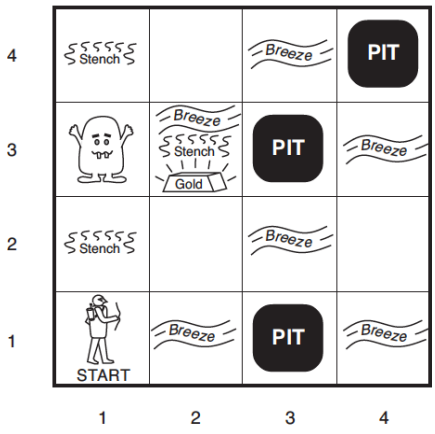
Performance Measure

- +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.

Environment

4x4 grid of rooms. Square [1,1] is initial state with agent facing to the right. Locations of gold, 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

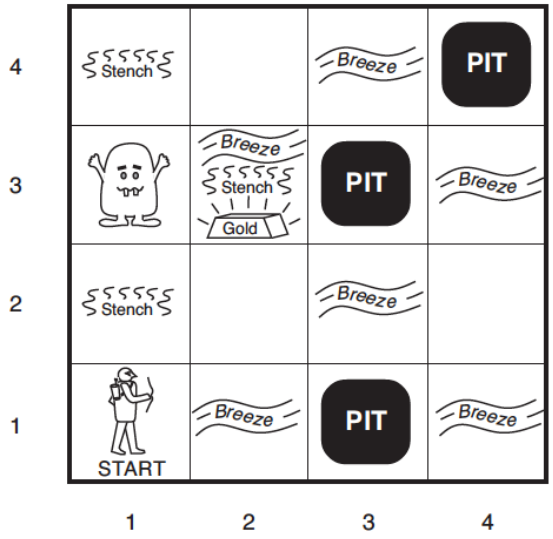
- The agent can move forward, turn right or left by 90 degrees
- Grab can be used to pick up an object in the same square as the agent.
- Shoot can be used to shoot the single arrow in a straight line until it hits something (Wumpus or a boundary wall)

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

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 A OK	2,1 OK	3,1	4,1

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

An Example(1)

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 A OK	2,1 OK	3,1	4,1

In $Rm_{1,1}$, there is no breeze or stench: $(\sim B_{1,1} \ \& \ \sim S_{1,1})$
Consequently, $Rm_{2,1}$ and $Rm_{1,2}$ are safe: $(OK_{2,1} \ \& OK_{1,2})$.

KB

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

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

An Example(2)

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

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 A B OK	3,1 P?	4,1

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,2}$ $B_{2,1}$, KB

What can A conclude about pits in its vicinity?

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,2}$ $B_{2,1}$, , $(P_{2,2} \mid P_{3,1})$ KB

Partial Observability as disjunctive information

Artificial Intelligence & Integrated Computer Systems Division
Department of Computer and Information Science
Linköping University, Sweden

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

An Example(3)

Since there may be a Pit in either $Rm_{2,2}$ or $Rm_{3,1}$: $(P_{2,2} \mid P_{3,1})$, A decides to move back to $Rm_{1,1}$ and then to $Rm_{1,2}$.

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

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1}$, , $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$ KB

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

Artificial Intelligence & Integrated Computer Systems Division
Department of Computer and Information Science
Linköping University, Sweden

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

An Example(4)

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$

KB

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{1,3})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$

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 $Rm_{2,2}$: $(\sim W_{2,2})$

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,} B_{2,1,}$, $(P_{2,2} \mid P_{1,3})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$

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

An Example(5)

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,} B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$

KB

But $(W_{1,3} \mid W_{2,2})$ and $(\sim W_{2,2})$ imply $W_{1,3}$, so there is a Wumpus in $R_{1,3}$

$$\frac{(W_{1,3} \mid W_{2,2}) \qquad \sim W_{2,2}}{W_{1,3}}$$ Resolution

$$\frac{\sim W_{2,2} \qquad \sim W_{2,2} \rightarrow W_{1,3}}{W_{1,3}}$$ Modus Ponens

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

An Example(6)

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$

KB

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

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 $Rm_{2,2}$: $(\sim P_{2,2})$

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$

But $(P_{2,2} \mid P_{3,1})$ and $(\sim P_{2,2})$ imply $P_{3,1}$, so there is a Pit in $R_{3,1}$

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$

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

An Example(7)

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$

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

Since it was inferred that there is no pit or wumpus in $Rm_{2,2}$ $(\sim P_{2,2}$ & $\sim W_{2,2})$, A can infer that Rm 2,2 is Ok $(OK_{2,2})$ and move there.

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,}$ $B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$, $OK_{2,2}$

What about $Rm_{2,3}$ *and* $Rm_{3,2}$?

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

An Example(8)

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

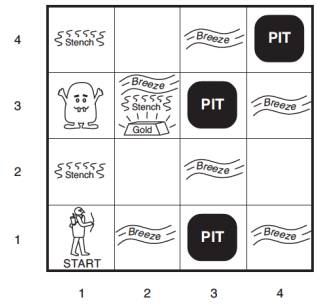
$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,} B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$, $OK_{2,2}$

Since there is no stench or breeze in $Rm_{2,2}$, Both $Rm_{2,3}$ and $Rm_{3,2}$ are ok to move to: ($OK_{2,3}$ & $OK_{3,2}$)

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,} B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$, $OK_{2,2,}$ $OK_{2,3}$, $OK_{3,2}$

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

An Example(9)



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	4,2
1,1 V OK	2,1 B V OK	3,1 P!	4,1

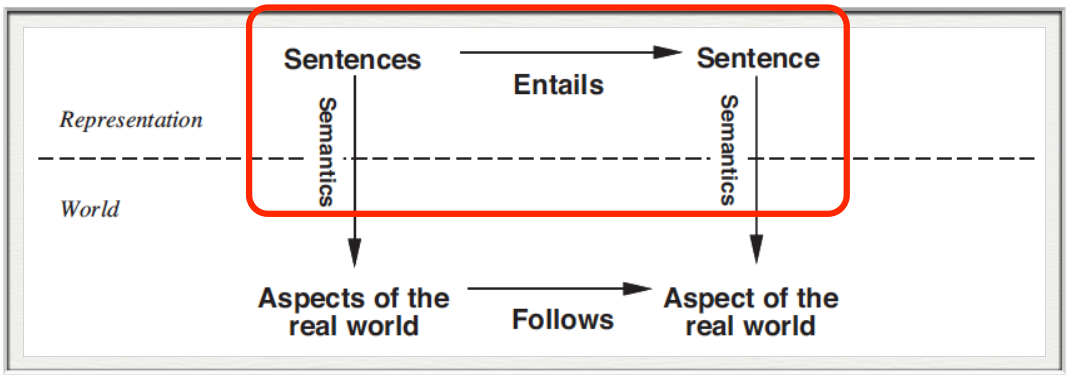
$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,} B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$, $OK_{2,2,}$ $OK_{2,3}$, $OK_{3,2}$

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{1,1}$, $OK_{2,1}$, $OK_{1,} B_{2,1,}$, $(P_{2,2} \mid P_{3,1})$, $S_{1,2}$, $(W_{1,3} \mid W_{2,2})$, $\sim W_{2,2}$, $\sim P_{2,2}$ $P_{3,1}$, $OK_{2,2,}$ $OK_{2,3}$, $OK_{3,2}$, $B_{3,2}$, $S_{3,2}$, $G_{3,2}$

A picks up the gold and wins the game!

Logic as a Representation Language



Propositional Logic

First-Order Logic

Default Nonmonotonic Reasoning

Resolution Theorem Proving

Resolution Theorem Proving

Language for 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: ∧, ∨, ⊃, and, ¬, called "and", "or", "implies", and "not".

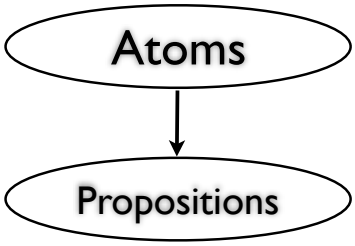
Syntax of well-formed formulas (wffs), also called ***sentences***:

- Any atom is a wff
- if ω1, ω2 are wffs, so are
- ω1 ∧ ω2 (conjunction)
- ω1 ∨ ω2 (disjunction)
- ω1 ⊃ ω2 (implication)
- ¬ω1 (negation)

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

Semantics

What do sentences *mean*?

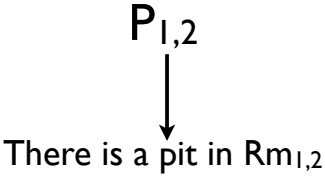


Semantics is about associating elements of a logical language with elements of a domain of discourse.

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 a proposition with each atom and a value (True or False)



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

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

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.

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 wffs) is called a **model** of the wff (set of wffs).

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

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

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

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 $\Delta \models \omega$

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

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

$F \models \omega$ (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

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 A B OK	3,1 P?	4,1

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

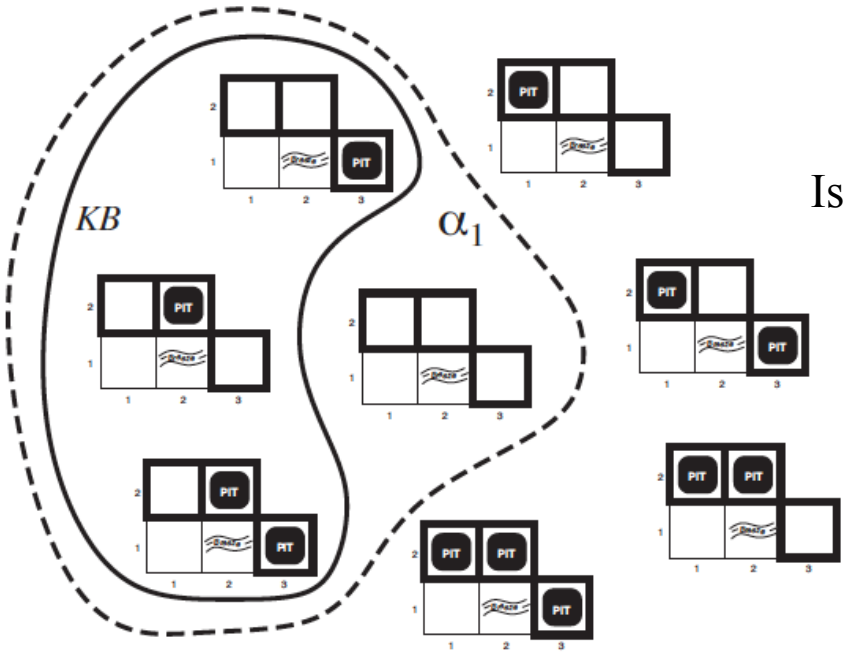
We want to reason about PITS in:
[1,2],[2,2],[3,1]

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

$\sim B_{1,1}$, $\sim S_{1,1}$, $OK_{2,1}$, $OK_{1,2}$
$B_{2,1}$, , $(P_{2,2} \mid P_{3,1})$

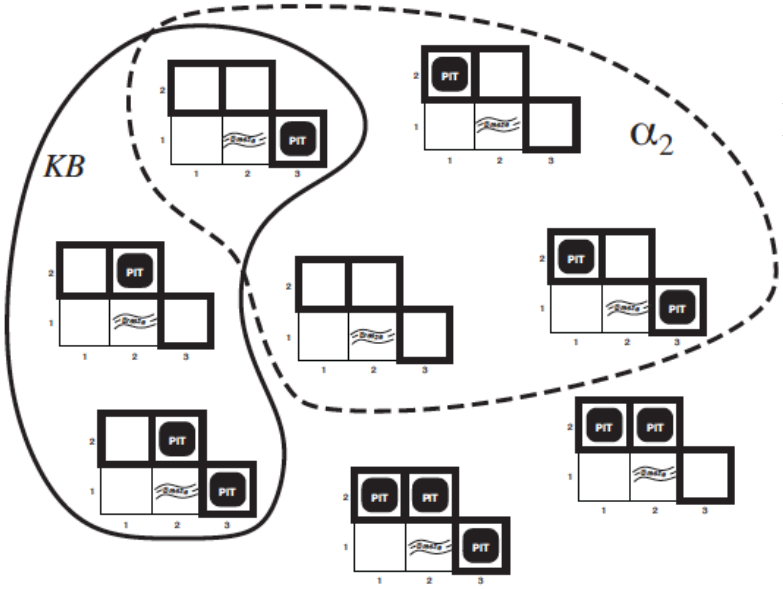
Wumpus Possible Worlds



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

Is α_1 entailed by KB?

Wumpus Possible Worlds



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

Is α_2 entailed by KB?

Truth Table Enumeration Algorithm

Entailment checking by enumeration

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($KB, \alpha, symbols, \{\}$)

---

function TT-CHECK-ALL($KB, \alpha, symbols, model$) **returns** *true* or *false*
  if EMPTY?($symbols$) **then**
    if PL-TRUE?($KB, model$) **then return** PL-TRUE?($\alpha, model$)
    else return *true* // *when KB is false, always return true*
  else do
    $P \leftarrow$ FIRST($symbols$)
    $rest \leftarrow$ REST($symbols$)
    return (TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = true\}$)
      and
      TT-CHECK-ALL($KB, \alpha, rest, model \cup \{P = false\}$))

Axiomatizing the Wumpus World

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

Schemas:

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

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

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

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

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.

Some Examples:

$$\frac{\omega 1,\ \omega 2}{\omega 1 \wedge \omega 2} \qquad \frac{\omega 1,\ \omega 1 \supset \omega 2}{\omega 2}$$

Definition of a Proof

The sequence of wffs {ω1, ω2, . . ., ωn} 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).

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:

$\Delta \vdash \omega_n$ $\quad (\Delta \vdash_{\Re} \omega_n$, *where* $\Re$ *refers to a set of inference rules*)

Proof of $Q \wedge R$ from Δ

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

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

Equivalence and Laws

Two wffs are said to be **equivalent** if and only if their truth values are identical under all interpretations. Equivalence is written using a ≡ sign.

DeMorgan's Laws

$$\neg(\omega1 \vee \omega2) \equiv \neg\omega1 \wedge \neg\omega2$$
$$\neg(\omega1 \wedge \omega2) \equiv \neg\omega1 \vee \neg\omega2$$

Law of the Contrapositive

$$\omega1 \supset \omega2 \equiv \neg\omega2 \supset \neg\omega1$$

Soundness and Completeness

If, for any set of wffs, Δ , and wff, ω, Δ |--$_{\Re}$ ω implies Δ |= ω , we say that the set of inference rules, $\Re$, is **sound**.

If, for any set of wffs, Δ , and wff, ω, it is the case that whenever Δ |= ω, there exists a proof of ω from Δ using the set of inference rules, $\Re$, we say that $\Re$ is **complete**.

Syntactic characterizations of Entailment

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

Important Meta Theorems

The Deduction Theorem

if $\{\omega 1, \omega 2, \ldots, \omega n\}$ |= ω then $(\omega 1 \wedge \omega 2 \wedge \ldots \omega n) \supset \omega$ 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 Δ |= ω

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

Clause Forms for Sentences

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

A **clause** is a set of literals. The set is an abbreviation for the disjunction of literals in the set.

$\{P, Q, \neg R\}$ $\quad P \vee Q \vee \neg R$ $\quad \{\ \}$

The empty clause is equivalent to F whose value is False.

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

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

Converting to CNF and DNF

1. Eliminate implication connectives $\supset$ by using the equivalent forms using $\vee$.
2. Reduce the scope of $\neg$ connectives by using DeMorgan's laws and by eliminating double $\neg$ connectives.
3. Convert to CNF by using the associative and distributive laws.

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

$\neg\ [((P \vee \neg Q) \supset R) \supset (R \wedge P)\ ]$

1. $\neg\ (\neg\ P \vee Q) \vee (\neg\ R \vee P)$
2. $(P \wedge \neg\ Q) \vee (\neg\ R \vee P)$
3. $(P \vee \neg\ R \vee P) \wedge (\neg\ Q \vee \neg\ R \vee P)$
   $(P \vee \neg\ R\ ) \wedge (\neg\ Q \vee \neg\ R \vee P)$

$\{(P \vee \neg\ R\ )\ ,\ (\neg\ Q \vee \neg\ R \vee P)\ \}$

A Rule of Inference: Resolution

$$\frac{\{\lambda\} \cup \Sigma_1 \qquad \{\neg\lambda\} \cup \Sigma_2}{\Sigma_1 \cup \Sigma_2}$$

λ is an atom and Σ_1 and Σ_2 are sets of literals.

(representing disjunctions)!

The process of resolving upon atoms is called **Resolution**.

The atom λ is the atom resolved upon.

$\Sigma_1 \cup \Sigma_2$ is called the **resolvent** of the 2 clauses.

$$\frac{(\neg P \vee Q) \quad (\neg R \vee P)}{(Q \vee \neg R)}$$

If $(P \supset Q)$ and $(R \supset P)$ then $(R \supset Q)$

Chaining is a special case of resolution

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

Resolving on a positive (λ)and negative ($\neg\lambda$) literal results in the empty clause { } (F). A contradiction.

Soundness of Resolution

$$\frac{\{\lambda\} \cup \Sigma_1 \qquad \{\neg\lambda\} \cup \Sigma_2}{\Sigma_1 \cup \Sigma_2}$$

The Resolution Inference Rule is sound:

Truth preservation

If $\{\lambda\} \cup \Sigma_1$ and $\{\neg\lambda\} \cup \Sigma_2$ are *True* then $\Sigma_1 \cup \Sigma_2$ is *True* also.

Δ |--$_{\Re}$ ω implies Δ |= ω, where ω is a clause.

Resolution is not Complete!

The Resolution Inference Rule is not complete:

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

$P \wedge R \models P \vee R$, but it is not the case that $P \wedge R \vdash_{\Re} P \vee R$

$\Delta \models \omega$ does **not** imply $\Delta \vdash_{\Re} \omega$, where ω is a clause.

Resolution can not be used directly to decide all logical entailments! **but . . .**

Resolution Refutation is Complete

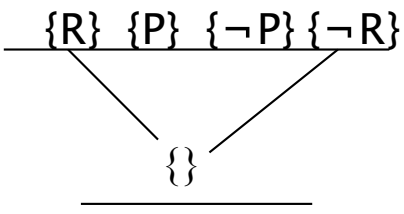
Remember our Meta Theorem!

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 using resolution that the negation of $P \vee R$ is inconsistent with $P \wedge R$:



Resolve on P

or

Resolve on R

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.

Using Resolution in the Wumpus World

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 KB ∪ $\{P_{1,2}\}$ is inconsistent

KB ∪ $\{P_{1,2}\}$ in CNF Form:

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

PL-RESOLUTION

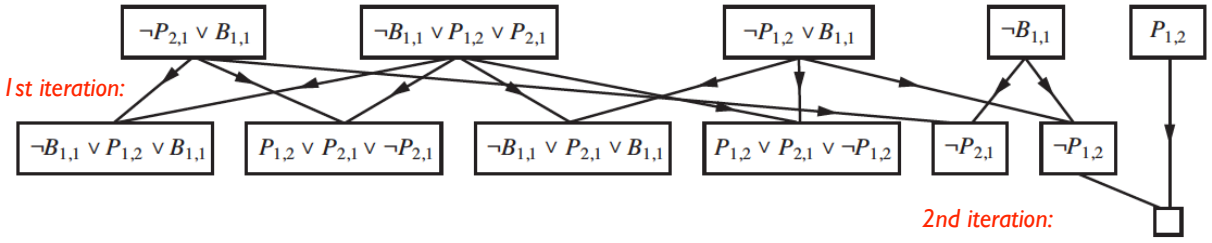
function PL-RESOLUTION(KB, α) **returns** *true* or *false*
  inputs: KB, the knowledge base, a sentence in propositional logic
    α, the query, a sentence in propositional logic

  clauses $\leftarrow$ the set of clauses in the CNF representation of $KB \wedge \neg\alpha$
  new $\leftarrow \{\}$
  loop do
    for each pair of clauses C_i, C_j **in** *clauses* **do**
      resolvents $\leftarrow$ PL-RESOLVE(C_i, C_j)
      if *resolvents* contains the empty clause **then return** *true*
      new $\leftarrow$ *new* $\cup$ *resolvents*
    if *new* $\subseteq$ *clauses* **then return** *false*
    clauses $\leftarrow$ *clauses* $\cup$ *new*

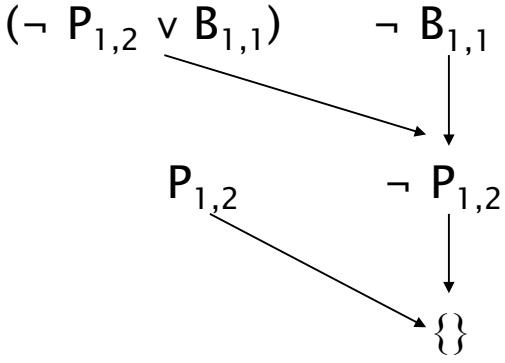
No new clauses generated, no empty clause

Using Resolution in the Wumpus World

Algorithm:



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



Logical Wumpus 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 ← {[x,y] : ASK(KB, OK^t_{x,y}) = true}
  if ASK(KB, Glitter^t) = true then
    plan ← [Grab] + PLAN-ROUTE(current,{[1,1]},safe) + [Climb]
  if plan is empty then
    unvisited ← {[x,y] : ASK(KB, L^{t'}_{x,y}) = false for all t' ≤ t}
    plan ← PLAN-ROUTE(current,unvisited ∩ safe,safe)
  if plan is empty and ASK(KB, HaveArrow^t) = true then
    possible_wumpus ← {[x,y] : ASK(KB, ¬ W_{x,y}) = false}
    plan ← PLAN-SHOT(current,possible_wumpus,safe)
  if plan is empty then   // no choice but to take a risk
    not_unsafe ← {[x,y] : ASK(KB, ¬ OK^t_{x,y}) = false}
    plan ← PLAN-ROUTE(current,unvisited ∩ not_unsafe,safe)
  if plan is empty then
    plan ← PLAN-ROUTE(current,{[1,1]},safe) + [Climb]
  action ← POP(plan)
  TELL(KB,MAKE-ACTION-SENTENCE(action,t))
  t ← t + 1
  return action
```

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 ← ROUTE-PROBLEM(current,goals,allowed)
  return A*-GRAPH-SEARCH(problem)
```

Effective Propositional Model Checking

Algorithms for general propositional inference:

- Backtracking Search
- Local Hill-Climbing Search

Entailment and Satisfiability

Checking that: $\alpha \models \beta$

Can be done by checking that: $\alpha \wedge \neg\beta$ is unsatisfiable

The Deduction Theorem

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

If α is valid, then $\neg\alpha$ is unsatisfiable

If $\alpha \rightarrow \beta$ is valid, then $\neg(\alpha \rightarrow \beta)$ is unsatisfiable

$$\neg(\alpha \rightarrow \beta) \equiv \alpha \wedge \neg\beta$$

Davis Putman 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
- 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

DPLL Algortihm

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.

Early Termination

If A is true in an assignment then

$$(A \vee B \vee D) \wedge (A \vee \neg E \vee F) \wedge (A \vee G)$$

is true without knowing assignment of other variables

If A, B, D are false in an assignment then

$$(A \vee B \vee D) \wedge (A \vee \neg E \vee F) \wedge (A \vee G)$$

is false without knowing assignment of other variables

Find-Pure-Symbol

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 C is not pure

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

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

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

Find-Unit-Clause

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

$$A = \text{true}, B = \text{false}$$

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

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

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

Find-Unit-Clause

$$A = \text{true}, B = \text{false}$$

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

$C = \text{true}$

Note that $(\neg C \vee F)$

now becomes a unit clause and one can propagate

$F = \text{true}, G = \text{true}$

$$(D \vee E)$$

Modern Extensions

- *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…

Local Search Algorithms for SAT

- Studying 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 are full assignments
  - Children generated by flipping a variable's assignment
  - Evaluation function -
    - count the number of unsatisfied clauses
    - minimize
  - Can be many local minima
    - Use randomness to escape

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 flips allowed before giving up

  model ← a random assignment of *true*/*false* to the symbols in *clauses*
  for $i = 1$ **to** *max_flips* **do**
    if *model* satisfies *clauses* **then return** *model*
    clause ← a randomly selected clause from *clauses* that is false in *model*
    with probability *p* 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*

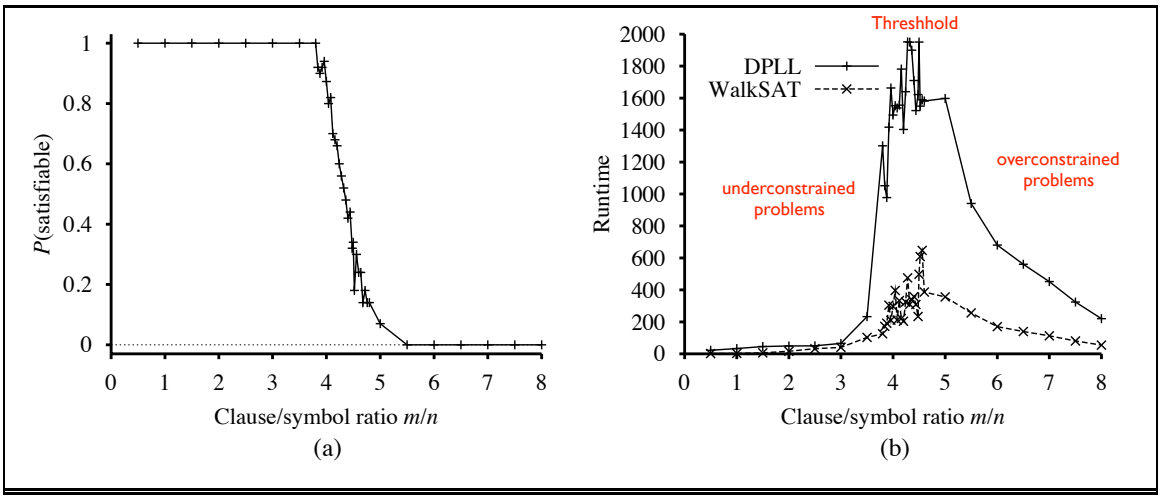
Two ways to choose the symbol to flip:

- *min-conflicts*: minimize the number of unsatisfied clauses
- *random walk*: pick the symbol randomly

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.

Some applications

- Kautz & Selman [1992] - "Planning as Satisfiability"
  - Classical Planning Problem could be solved by translating it into a propositional formula and testing its satisfiability
  - The plan itself could be extracted from a model that satisfies the resulting formula
  - In the simplest case, a classical planing problem consists of
    - an initial state
    - a set of actions
    - and a goal state
  - This approach to planning was later shown to be competitive with other approaches to planning
  - This idea led to attempts to encode other problems using these ideas
    - model checking in computer-aided software verification
    - SAT-based approach model-checking properties expressed in temporal logics
    - diagnosis and diagnosability testing for discrete even systems

SATPLAN: Planning and SAT

function SATPLAN($init$, $transition$, $goal$, $T_{\max}$) **returns** solution or failure
  inputs: $init$, $transition$, $goal$, constitute a description of the problem
    $T_{\max}$, an upper limit for plan length

  for $t = 0$ **to** $T_{\max}$ **do**
    $cnf \leftarrow$ TRANSLATE-TO-SAT($init$, $transition$, $goal$, t)
    $model \leftarrow$ SAT-SOLVER(cnf)
    if $model$ is not null **then**
      return EXTRACT-SOLUTION($model$)
  return $failure$