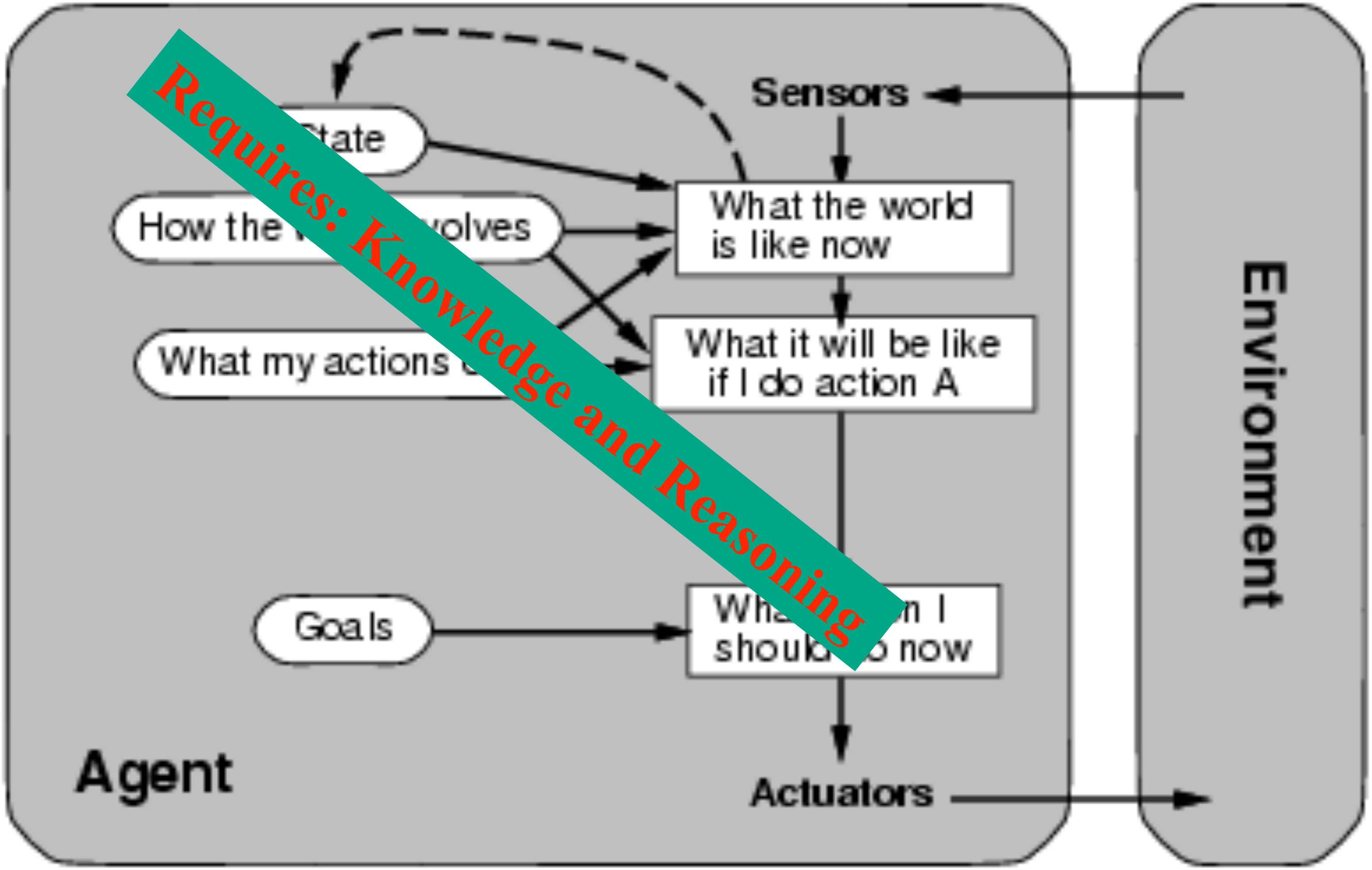
CS 4700: Foundations of Artificial Intelligence

Bart Selman
selman@cs.cornell.edu

Module: Knowledge, Reasoning, and Planning
Part 1

Logical Agents
R&N: Chapter 7

A Model-Based Agent



Knowledge and Reasoning

Knowledge and Reasoning:

humans are very good at acquiring new information by combining raw knowledge, experience with reasoning.

AI-slogan: "Knowledge is power" (or "Data is power"?)

Examples:

Medical diagnosis --- physician diagnosing a patient
infers what disease, based on the knowledge he/she acquired as a student, textbooks, prior cases

Common sense knowledge / reasoning ---
common everyday assumptions / inferences
e.g., "lecture starts at four" infer pm not am;
when traveling, I assume there is some way to get from the airport to the hotel.

Logical agents:

Agents with some representation of the complex knowledge about the world / its environment, and uses inference to derive new information from that knowledge combined with new inputs (e.g. via perception).

Key issues:

1- Representation of knowledge

What form? Meaning / semantics?

2- Reasoning and inference processes

Efficiency.

Knowledge-base Agents

Key issues:

- Representation of knowledge → knowledge base
- Reasoning processes → inference/reasoning

Knowledge base = set of sentences in a formal language representing facts about the world(*)

(*) called Knowledge Representation (KR) language

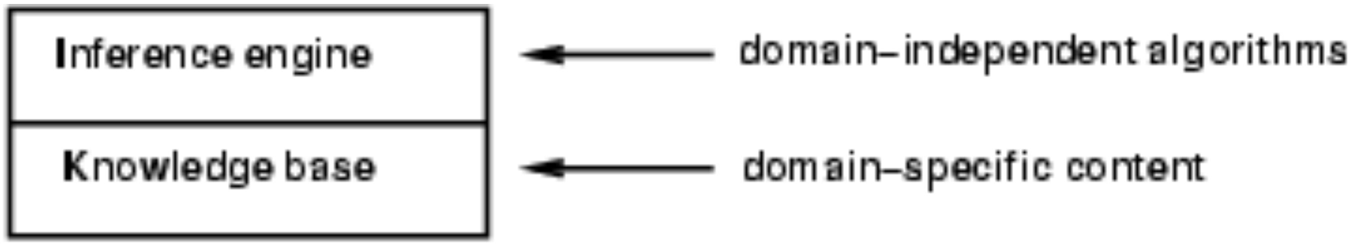
Knowledge bases

Key aspects:

- **How to add sentences to the knowledge base**
- **How to query the knowledge base**

Both tasks may involve inference – i.e. how to derive new sentences from old sentences

Logical agents – inference must obey the fundamental requirement that when one asks a question to the knowledge base, the answer should *follow* from what has been told to the knowledge base previously. (In other words the inference process should not "make things" up…)



A simple knowledge-based agent

```
function KB-Agent(percept) returns an action
    static: 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
```

The agent must be able to:

- **Represent states, actions, etc.**
- **Incorporate new percepts**
- **Update internal representations of the world**
- **Deduce hidden properties of the world**
- **Deduce appropriate actions**

KR language candidate:

logical language (propositional / first-order) combined with a logical inference mechanism

How close to human thought? (mental-models / Johnson-Laird).

What is "the language of thought"?

Greeks / Boole / Frege --- Rational thought: Logic?

Why not use natural language (e.g. English)?

We want clear syntax & semantics (well-defined meaning), and, mechanism to infer new information.
Soln.: Use a formal language.

"Advice-Taker"

1958 / 1968 — John McCarthy: "Programs with Common Sense" —
agents use logical reasoning to mediate between percepts and a

Idea: Impart knowledge to a program in the form of declarative (logical) statements ("what" instead of "how"); program uses general reasoning mechanisms to process and act on this information.

E.g. Formalize *"x is at y"* using predicate *at*, i.e., *at(x,y)*

at **defined** by its properties, e.g., $at(x, y) \wedge at(y, z) \rightarrow at(x, z)$

Problems??

Consider: to-the-right-of(x,y)

Agent / Intelligent System Design

Craik (1943) *The Nature of Explanation*

> If the organism carries a "small-scale model" of external reality and of its own small possible actions within its head, it is able to try out various alternatives, conclude which is the best of them, react to future situations before they arise, utilize the knowledge of the past events in dealing with the present and future, and in every way to react in a much fuller, safer, and more competent manner to the emergencies which face it.

Alt. view: against representations — Brooks (1989)

Representation Language

preferably:

— expressive and concise

— unambiguous and independent of context

— have an effective procedure to derive new information

not easy to meet these goals . . .

propositional and first-order logic meet some of the criteria

incompleteness / uncertainty is key — contrast with programming languages.

Procedural style:

```
printColor(snow) :- !, write("It's white.").
printColor(grass) :- !, write("It's green.").
printColor(sky) :- !, write("It's blue.").
printColor(X) :- write("Beats me.").
```

Knowledge-based alternative:

```
printColor(X) :-
     color(X,Y), !, write("It's "), write(Y), write("

color(snow,white).     (``KB'')
color(grass,green).
color(sky,yellow).
```

Modular.
Change KB without changing program.

Logical Representation

Three components:

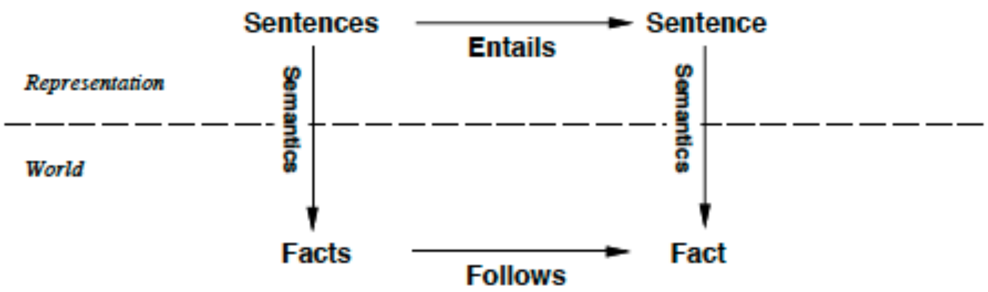
syntax

semantics (link to the world)

proof theory ("pushing symbols")

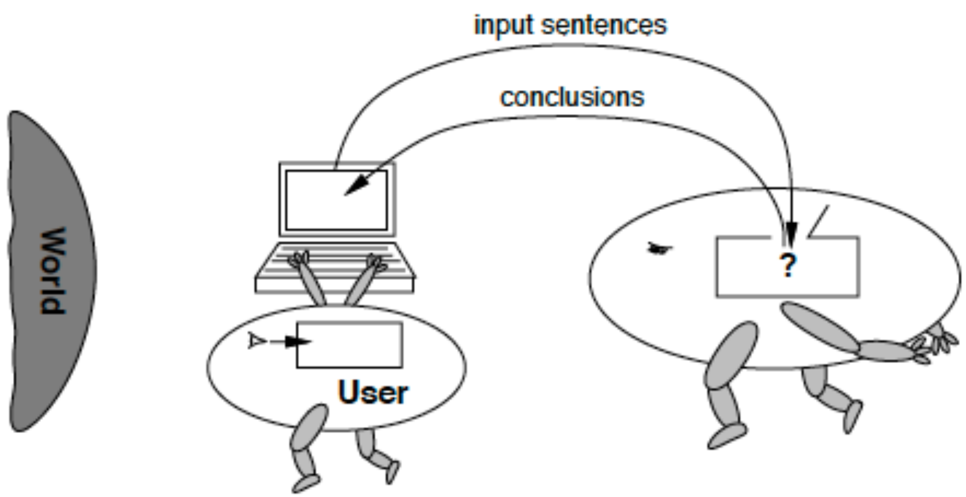
To make it work: **soundness** and **completeness**.

Connecting Sentences to the World



Somewhat misleading: formal semantics brings sentence down only to the primitive components (propositions). (later)

Tenuous Link to Real World



All computer has are sentences (hopefully about the world).

Sensors can provide some grounding.

Hope KB unique model / interpretation: the real-world.

Often many more... (Aside: consider arithmetic.)

The "symbol grounding problem."

More Concrete: Propositional Logic

Syntax: build sentences from atomic propositions, using connectives $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$.
(and / or / not / implies / equivalence (biconditional))

E.g.: $((\neg P) \vee (Q \wedge R)) \Rightarrow S$

Semantics

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*

Note: $\Rightarrow$ somewhat counterintuitive.

What's the truth value of "5 is even implies Sam is smart"?

True!

Validity and Inference

P	H	$P \vee H$	$(P \vee H) \wedge \neg H$	$((P \vee H) \wedge \neg H) \Rightarrow P$
False	*False*	*False*	*False*	*True*
False	*True*	*True*	*False*	*True*
True	*False*	*True*	*True*	*True*
True	*True*	*True*	*False*	*True*

Truth table for: $Premises \Rightarrow Conclusion.$

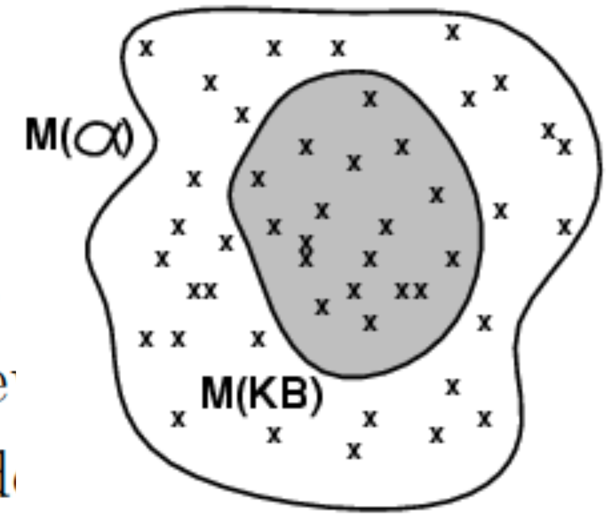
Shows $((P \vee H) \wedge (\neg H)) \Rightarrow P$ is valid

(True in all interpretations)

We write $\models ((P \vee H) \wedge (\neg H)) \Rightarrow P$

Compositional semantics

Models



A **model** of a set of sentences (KB) is a
in which each of the KB sentences e
With more and more sentences, the mod ng
more and more like the "real-world" (or isomorphic to it).

If a sentence α holds (is *True*) in **all** models of a KB, we say that α is **entailed** by the KB.

α is of interest, because *whenever KB is true in a world α will also be True.*

We write: $KB \models \alpha$.

Note: KB defines exactly the set of worlds we are interested in.

I.e.: **Models(KB)** $\subseteq$ **Models(** α **)**

Proof Theory

Purely syntactic rules for deriving the logical consequences o
a set of sentences.

We write: $KB \vdash \alpha$, i.e., α can be **deduced** from KB or α is **provable** from KB.

Key property:

Both in propositional and in first-order logic we have a proof theory ("calculus") such that:

$\vdash$ **and** $\models$ **are equivalent.**

Proof Theory

If $KB \vdash \alpha$ implies $KB \models \alpha$, we say the proof theory is **sound**.

If $KB \models \alpha$ implies $KB \vdash \alpha$, we say the proof theory is **complete**.

Why so remarkable / important?

Soundness and Completeness

Allows computer to ignore semantics and "just push symbols"!

In propositional logic, truth tables cumbersome (at least).

In first-order, models can be infinite!

Proof theory: One or more **inference rules** with

zero or more axioms (tautologies / to get things "going.").

**Note: (1) This was Aristotle's original goal ---
Construct infallible arguments based purely
on the *form of statements* --- not on the "meaning"
of individual propositions.
(2) Sets of models can be exponential size or worse,
compared to symbolic inference (deduction).**

Example Proof Theory

One rule of inference: **Modens Ponens**

From α and $\alpha \Rightarrow \beta$ it follows that β.

Semantic soundness easily verified. (truth table)

Axiom schemas:

(Ax. I) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

(Ax. II) $((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)))$.

(Ax. III) $(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\alpha \Rightarrow \neg\beta) \Rightarrow \alpha$.

Note: α, β, γ stand for arbitrary sentences. So, infinite collection of axioms.

Now, α can be **deduced** from a set of sentences Φ

iff there exists a sequence of applications of **modens ponens**

that leads from Φ to α (possibly using the axioms).

One can prove that:

Modens ponens with the above axioms will generate exactly

all (and only those) statements logically **entailed** by Φ.

So, we have a way of generating entailed statements

in a purely syntactic manner!

(Sequence is called a proof. Finding it can be hard ...)

(Ax. I) $\alpha \Rightarrow (\beta \Rightarrow \alpha)$

(Ax. II) $((\alpha \Rightarrow (\beta \Rightarrow \gamma)) \Rightarrow ((\alpha \Rightarrow \beta) \Rightarrow (\alpha \Rightarrow \gamma)))$.

(Ax. III) $(\neg\alpha \Rightarrow \beta) \Rightarrow (\neg\alpha \Rightarrow \neg\beta) \Rightarrow \alpha$.

Lemma. For any α, we have $\vdash (\alpha \Rightarrow \alpha)$.

Proof.

$(\alpha \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha) \Rightarrow (\alpha \Rightarrow \alpha \Rightarrow \alpha) \Rightarrow \alpha \Rightarrow \alpha$, (Ax. II)

$\alpha \Rightarrow (\alpha \Rightarrow \alpha) \Rightarrow \alpha$, (Ax. I)

$(\alpha \Rightarrow \alpha \Rightarrow \alpha) \Rightarrow \alpha \Rightarrow \alpha$, (M. P.)

$\alpha \Rightarrow \alpha \Rightarrow \alpha)$ (Ax. I)

$\alpha \Rightarrow \alpha$ (M.P.)

Illustrative example: Wumpus World

(Somewhat whimsical!)

Performance measure

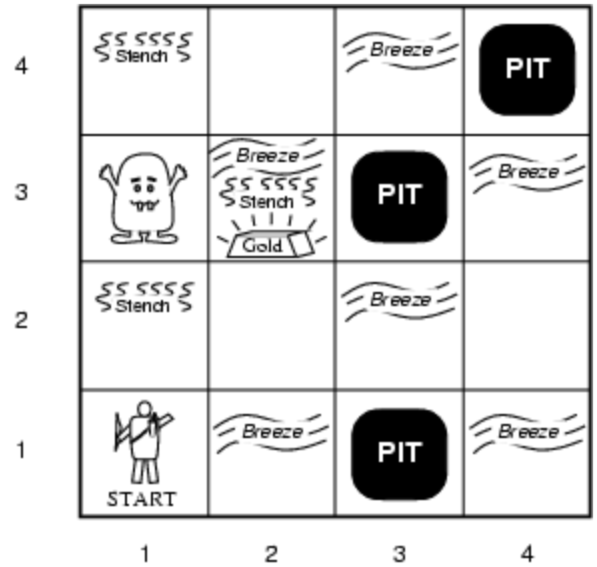
- gold +1000,
- death -1000

(falling into a pit or being eaten by the wumpus)

- -1 per step, -10 for using the arrow

Environment

- Rooms / squares connected by doors.
- 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
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square
- Randomly generated at start of game. Wumpus only senses current room.



Sensors: Stench, Breeze, Glitter, Bump, Scream [perceptual inputs]

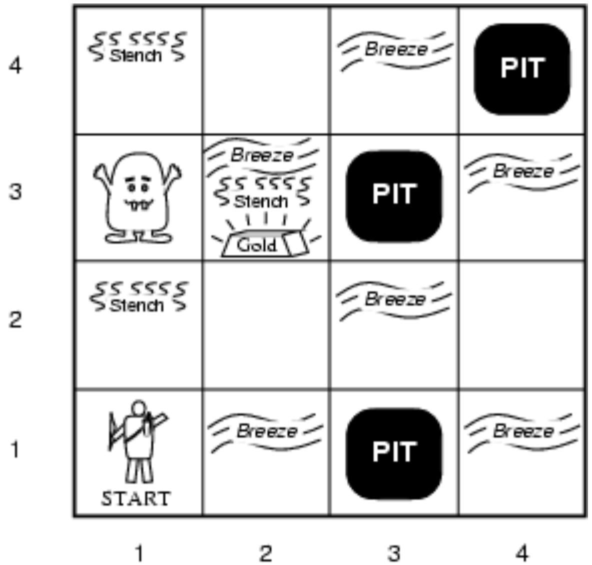
Actuators: Left turn, Right turn, Forward, Grab, Release, Shoot

Wumpus world characterization

Fully Observable	**No – only local perception**
Deterministic	**Yes – outcomes exactly specified**
Static	**Yes – Wumpus and Pits do not move**
Discrete	**Yes**
Single-agent?	**Yes – Wumpus is essentially a "natural feature."**

Exploring a wumpus world

The knowledge base of the agent consists of the rules of the Wumpus world plus the percept "nothing" in [1,1]

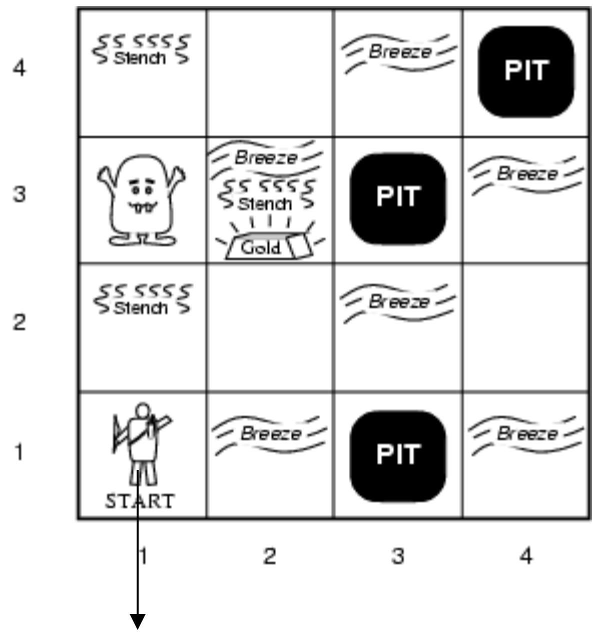


None, none, none, none, none

Stench, Breeze, Glitter, Bump, Scream

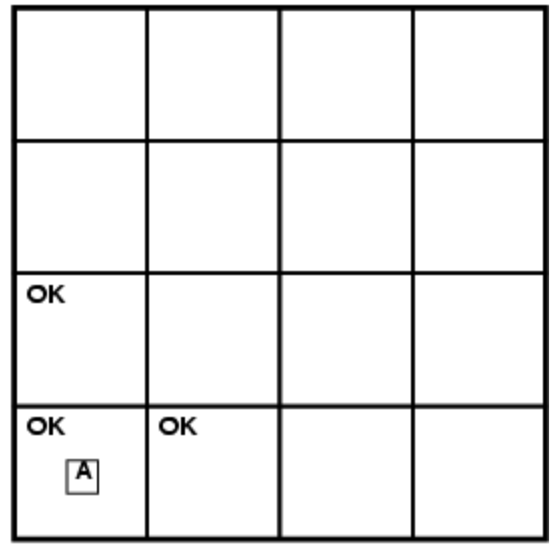
Boolean percept feature values:
<0, 0, 0, 0, 0>

World "known" to agent at time = 0.



None, none, none, none, none

Stench, Breeze, Glitter, Bump, Scream



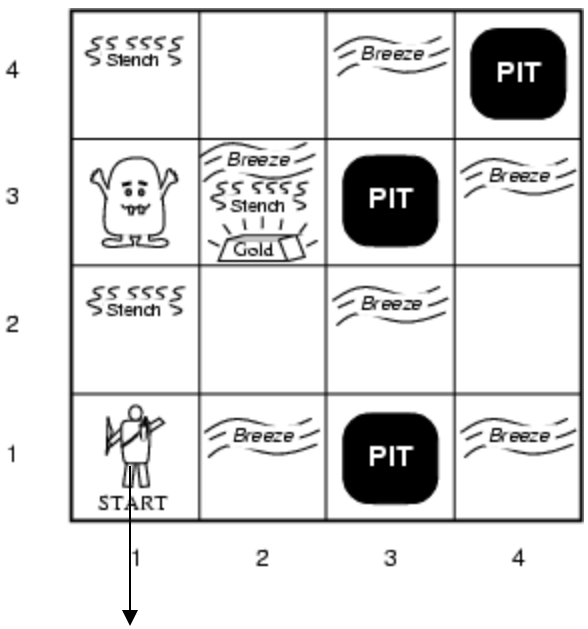
T=0 The KB of the agent consists of the rules of the Wumpus world plus the percept "nothing" in [1,1].

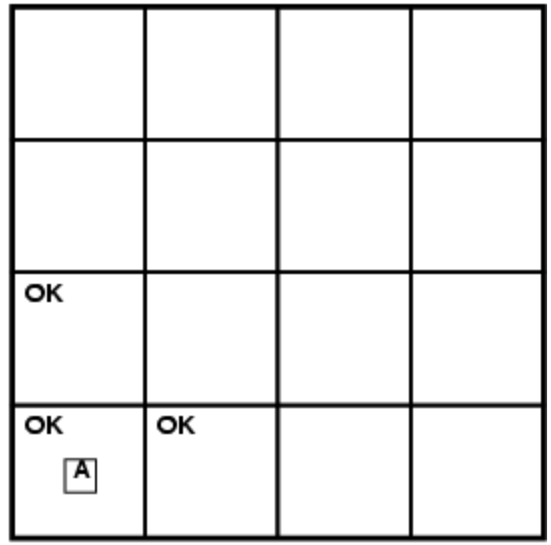
By inference, the agent's knowledge base also has the information that [2,1] and [1,2] are okay.

Added as propositions.

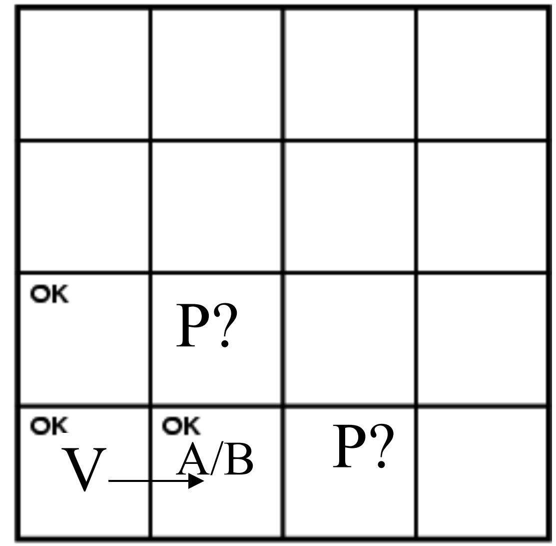
Further exploration

T = 0





T = 1



None, none, none, none, none

None, breeze, none, none, none

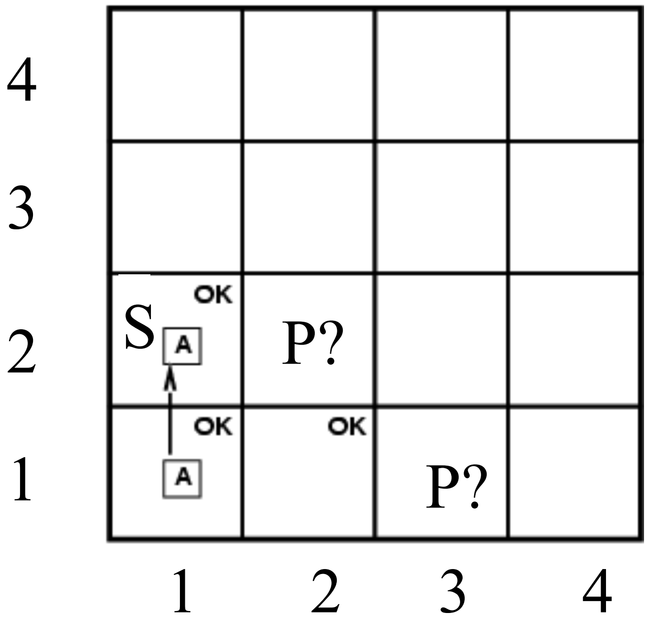
A – agent
V – visited
B - breeze

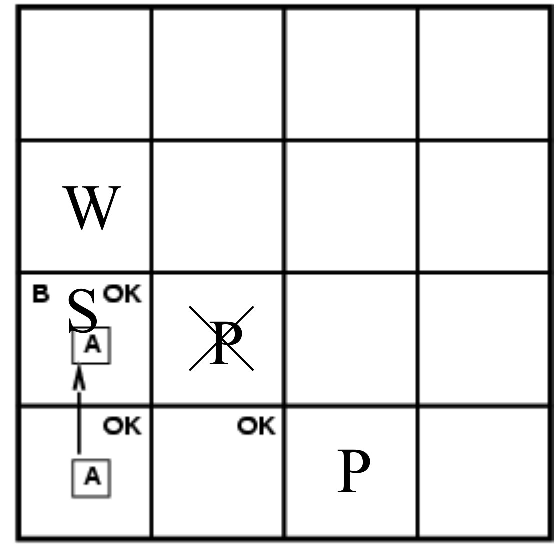
Stench, Breeze, Glitter, Bump, Scream

Where next?

@ T = 1 What follows?
Pit(2,2) or Pit(3,1)

T=3





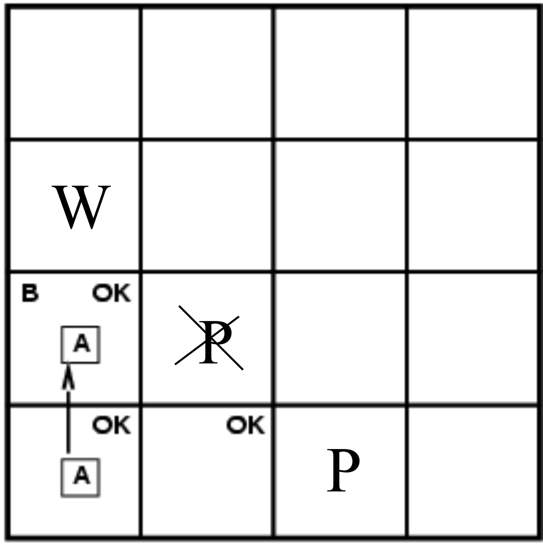
Stench, none, none, none, none

Stench, Breeze, Glitter, Bump, Scream

Where is Wumpus?

Wumpus cannot be in (1,1) or in (2,2) (Why?)➔ Wumpus in (1,3)
Not breeze in (1,2) ➔ no pit in (2,2); but we know there is
pit in (2,2) or (3,1) ➔ pit in (3,1)

We reasoned about the possible states the Wumpus world can be in, given our percepts and our knowledge of the rules of the Wumpus world.
I.e., the content of KB at T=3.



What follows is what holds true in all those worlds that satisfy what is known at that time T=3 about the particular Wumpus world we are in.

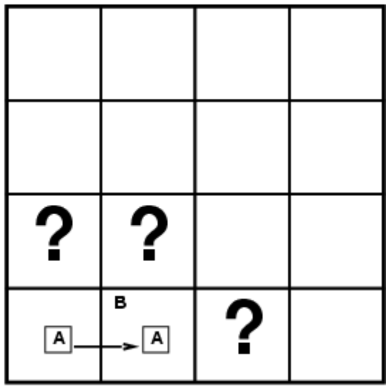
Example property: P_in_(3,1)

Models(KB) $\subseteq$ Models(P_in_(3,1))

Essence of logical reasoning:
Given *all we know*, Pit_in_(3,1) holds.
("The world cannot be different.")

Formally: Entailment

Knowledge Base (KB) in the Wumpus World → Rules of the wumpus world + new percepts



T = 1

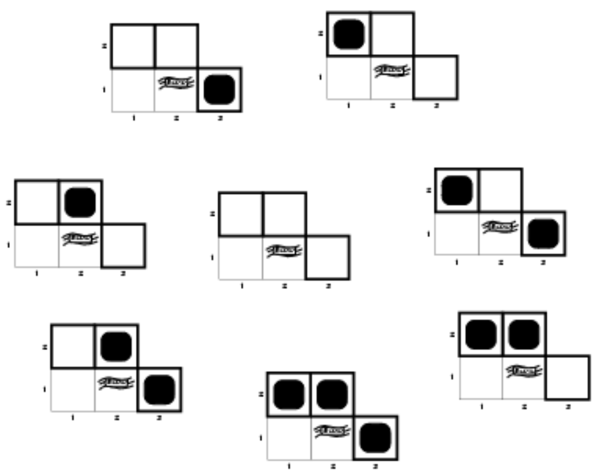
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]. I.e. T=1.

Consider possible models for *KB* with respect to the cells (1,2), (2,2) and (3,1), with <u>respect to the existence or non existence of pits</u>

3 Boolean choices ⇒

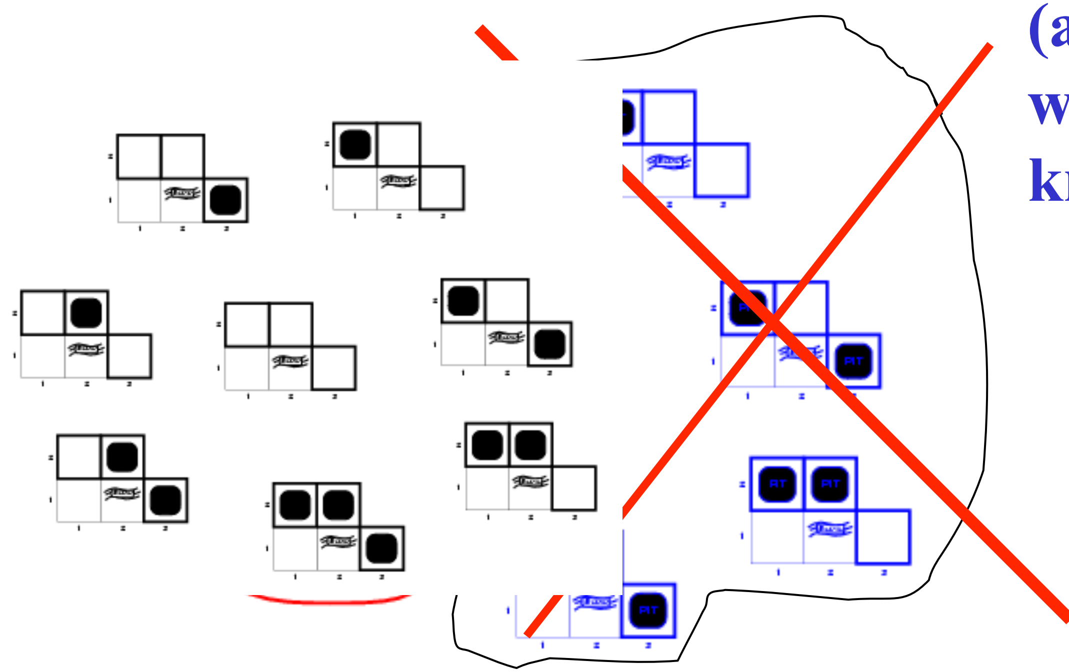
8 possible interpretations

(enumerate all the models or "possible worlds" wrt Pitt location)

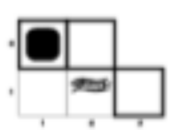


Is KB consistent with all 8 possible worlds?

Worlds that violate KB (are inconsistent with what we know)



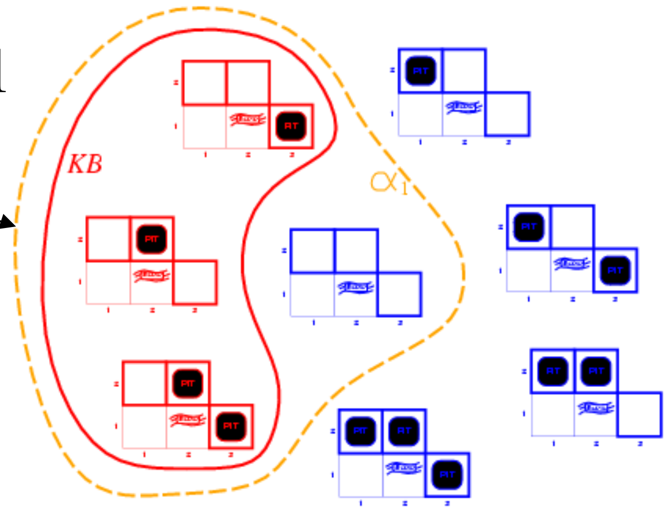
KB **= Wumpus-world rules + observations (T=1)**

Q: Why does world  violate KB?

Entailment in Wumpus World

So, KB defines all worlds that we hold possible.
Queries: we want to know the properties of those worlds.
That's how the semantics of logical entailment is defined.

Models of the KB and α1



Note: \alpha_1 holds in more models than KB. That's OK, but we don't care about those worlds.

KB = Wumpus-world rules + observations

α_1 = "[1,2] has no pit", $KB \models \alpha_1$

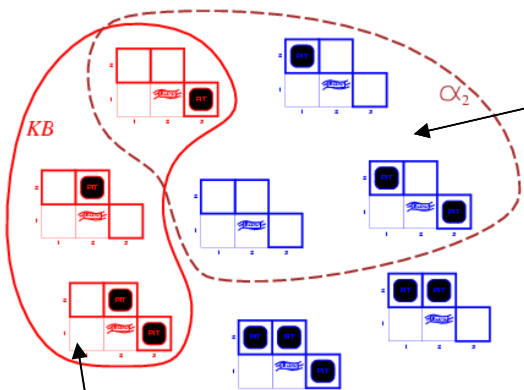
- **In every model in which KB is true, α_1 is True (proved by "model checking")**

Wumpus models

KB = wumpus-world rules + observations

α2 = "[2,2] has no pit", this is only True in some of the models for which KB is True, therefore KB $\not\models$ α2

Model Checking



Models of α2

A model of KB where α2 does NOT hold!

Entailment via "Model Checking"

Inference by Model checking –

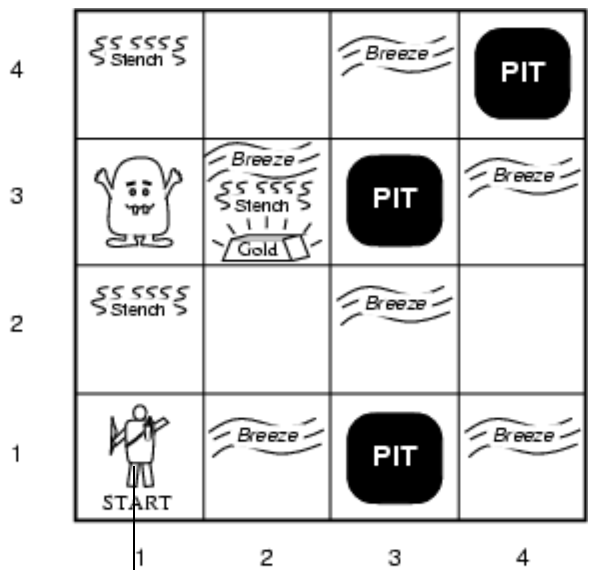
We enumerate all the KB models and check if α_1 and α_2 are True in all the models (which implies that we can only use it when we have a finite number of models).

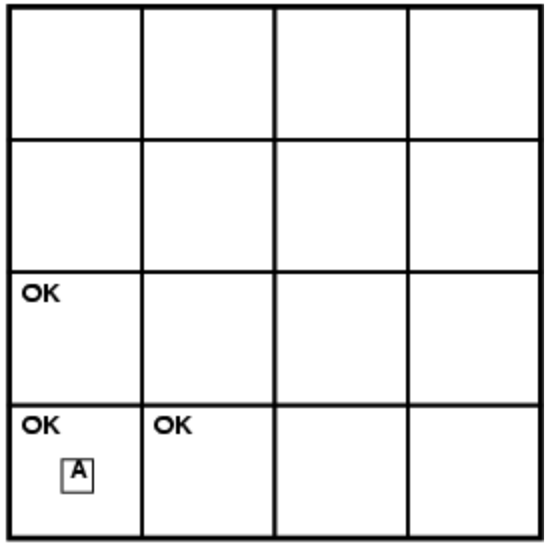
I.e. using semantics directly.

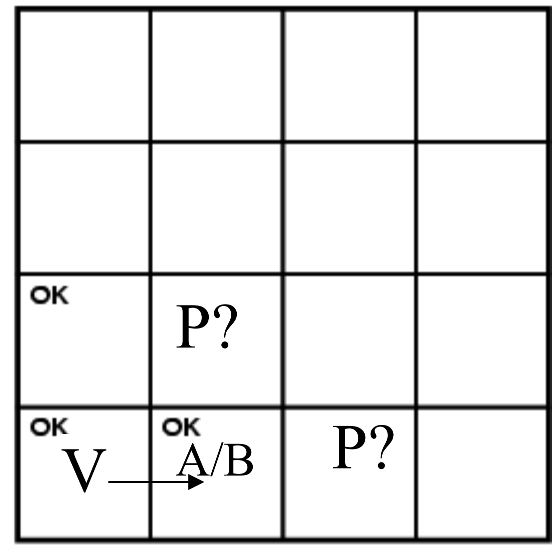
$$\text{Models(KB)} \subseteq \text{Models}(\alpha)$$

$$KB \models \alpha$$

Example redux: More formal







None, none, none, none, none

Stench, Breeze, Glitter, Bump, Scream

None, breeze, none, none, none

A – agent
V – visited
B - breeze

How do we actually encode background knowledge and percepts in formal language?

Wumpus World KB

Define propositions:

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].

Sentence 1 (R1): $\neg P_{1,1}$ [Given.]

Sentence 2 (R2): $\neg B_{1,1}$ [Observation T = 0.]

Sentence 3 (R3): $B_{2,1}$ [Observation T = 1.]

"Pits cause breezes in adjacent squares"

Sentence 4 (R4): $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

Sentence 5 (R5): $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

etc.

Notes: (1) one such statement about Breeze for each square.

(2) similar statements about Wumpus, and stench and Gold and glitter. (Need more propositional letters.)

What about Time? What about Actions?

Is Time represented?

No!

Can include time in propositions:

- **Explicit time** $P_{i,j,t}$ $B_{i,j,t}$ $L_{i,j,t}$ **etc.**
- **Many more props:** $O(TN^2)$ ($L_{i,j,t}$ for agent at (i,j) at time t)

Now, we can also model actions, use props: Move(i ,j ,k ,l ,t)

- **E.g. Move(1, 1, 2, 1, 0)**

What knowledge axiom(s) capture(s) the effect of an Agent move?

$$Move(i, j, k, l, t) \Rightarrow (\neg L(i, j, t+1) \wedge L(k, l, t+1))$$

Is this it?

What about i, j, k, and l?

What about Agent location at time t?

Improved: *Move implies a change in the world state;*
a change in the world state, implies a move occurred!

Move(i, j, k, l ,t) $\Leftrightarrow$ (L(i, j, t) $\wedge$ $\neg$ L(i, j, t+1) $\wedge$ L(k, l, t+1))

For all tuples (i, j, k, l) that represent legitimate possible moves.

E.g. (1, 1, 2, 1) or (1, 1, 1, 2)

Still, some remaining subtleties when representing time and actions. What happens to propositions at time t+1 compared to at time t, that are *not* involved in any action?

E.g. P(1, 3, 3) is derived at some point.

What about P(1, 3, 4), True or False?

R&N suggests having P as an "atemporal var" since it cannot change over time. Nevertheless, we have many other vars that can change over time, called "fluents".

Values of propositions not involved in any action should not change! "The Frame Problem" / Frame Axioms R&N 7.7.1

Successor-State Axioms

Axiom schema:

F is a fluent (prop. that can change over time)

For example:

$$L_{1,1}^{t+1} = (L_{1,1}^{t} \wedge (\neg Forward^{t} \vee Bump^{t+1}))$$
$$\vee (L_{1,2}^{t} \wedge (South^{t} \wedge Forward^{t}))$$
$$\vee (L_{2,1}^{t} \wedge (West^{t} \wedge Forward^{t}))$$

i.e. L_1,1 was "as before" with [no movement action or bump into wall] or resulted from some action (movement into L_1,1).

Actions and inputs up to time 6

Note: includes turns!

Some example inferences
Section 7.7.1 R&N

$$\neg Stench^0 \land \neg Breeze^0 \land \neg Glitter^0 \land \neg Bump^0 \land \neg Scream^0 \; ; \; Forward^0$$
$$\neg Stench^1 \land Breeze^1 \land \neg Glitter^1 \land \neg Bump^1 \land \neg Scream^1 \; ; \; TurnRight^1$$
$$\neg Stench^2 \land Breeze^2 \land \neg Glitter^2 \land \neg Bump^2 \land \neg Scream^2 \; ; \; TurnRight^2$$
$$\neg Stench^3 \land Breeze^3 \land \neg Glitter^3 \land \neg Bump^3 \land \neg Scream^3 \; ; \; Forward^3$$
$$\neg Stench^4 \land \neg Breeze^4 \land \neg Glitter^4 \land \neg Bump^4 \land \neg Scream^4 \; ; \; TurnRight^4$$
$$\neg Stench^5 \land \neg Breeze^5 \land \neg Glitter^5 \land \neg Bump^5 \land \neg Scream^5 \; ; \; Forward^5$$
$$Stench^6 \land \neg Breeze^6 \land \neg Glitter^6 \land \neg Bump^6 \land \neg Scream^6$$

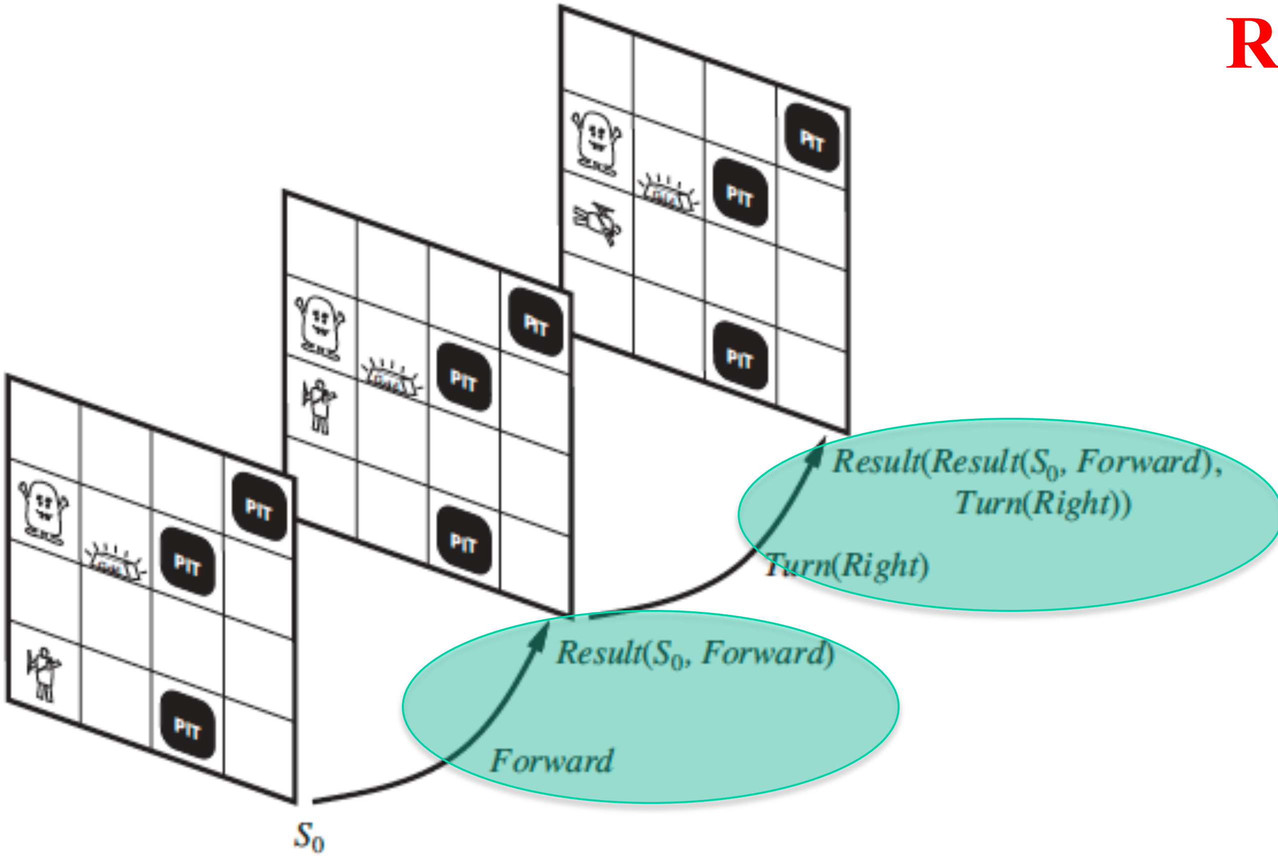
$\textsc{Ask}(KB, P_{3,1}) = true$ $\quad$ $\textsc{Ask}(KB, W_{1,3}) = true$

Define "OK": $OK^t_{x,y} \Leftrightarrow \neg P_{x,y} \land \neg(W_{x,y} \land WumpusAlive^t)$

$\textsc{Ask}(KB, OK^6_{2,2}) = true$, so the square [2, 2] is OK

In milliseconds, with modern SAT solver.

Alternative formulation: Situation Calculus R&N 10.4.2



No explicit time. Actions are what changes the world from "situation" to "situation". More elegant, but still need frame axioms to capture what stays the same. Inherent with many representation formalisms: "physical" persistance does not come for free! (and probably shouldn't)

Inference by enumeration / "model checking" Style I

The goal of logical inference is to decide whether $KB \models \alpha$, for some α.

For example, given the rules of the Wumpus World, is P_{22} entailed? Relevant propositional symbols:

R1: $\neg P_{1,1}$

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

"Pits cause breezes in adjacent squares"

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

$$\text{Models(KB)} \overset{?}{\subseteq} \text{Models(P22)}$$

Inference by enumeration. We have 7 relevant symbols
Therefore $2^7 = 128$ interpretations.
Need to check if P22 is true in all of the KB models (interpretations that satisfy KB sentences).

Q.: KB has many more symbols. Why can we restrict ourselves to these symbols here? But, be careful, typically we can't!!

All equivalent
Prop. / FO Logic

1) $KB \models \alpha$ — **entailment**

2) $M(KB) \subseteq M(\alpha)$ — **by defn. / semantic proofs / truth tables**
"model checking" /enumeration
(style I, R&N 7.4.4)

3) $\models (KB \Rightarrow \alpha)$ — **deduction thm. R&N 7.5**

4) $KB \vdash \alpha$ — **soundness and completeness**
logical deduction / symbol pushing
proof by inference rules (style II)
e.g. modus ponens (R&N 7.5.1)

5) $(KB \wedge \neg \alpha)$ is inconsistent — **Proof by contradiction**
use CNF / clausal form
Resolution (style III, R&N 7.5)
SAT solvers (style IV, R&N 7.6)
most effective

Proof techniques

$M(KB) \subseteq M(\alpha)$ — by defn. / semantic proofs / truth tables
"model checking"
(style I, R&N 7.4.4) Done.

$KB \vdash \alpha$ — soundness and completeness
logical deduction / symbol pushing
proof by inference rules (style II)
e.g. modus ponens (R&N 7.5.1)

$(KB \wedge \neg \alpha)$ is inconsistent — Proof by contradiction
use CNF / clausal form
Resolution (style III, R&N 7.5)
SAT solvers (style IV, R&N 7.6)
most effective

Aside

Standard syntax and semantics for propositional logic. (CS-2800; see 7.4.1 and 7.4.2.)

Syntax:

$$
\begin{aligned}
\mathit{Sentence} &\rightarrow \mathit{AtomicSentence} \mid \mathit{ComplexSentence} \\
\mathit{AtomicSentence} &\rightarrow \mathit{True} \mid \mathit{False} \mid P \mid Q \mid R \mid \ldots \\
\mathit{ComplexSentence} &\rightarrow (\ \mathit{Sentence}\ ) \mid [\ \mathit{Sentence}\ ] \\
&\mid\ \neg\ \mathit{Sentence} \\
&\mid\ \mathit{Sentence} \land \mathit{Sentence} \\
&\mid\ \mathit{Sentence} \lor \mathit{Sentence} \\
&\mid\ \mathit{Sentence} \;\Rightarrow\; \mathit{Sentence} \\
&\mid\ \mathit{Sentence} \;\Leftrightarrow\; \mathit{Sentence}
\end{aligned}
$$

OPERATOR PRECEDENCE : $\neg, \land, \lor, \Rightarrow, \Leftrightarrow$

Semantics

Note: Truth value of a sentence is built from its parts "compositional semantics"

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Logical equivalences

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \quad (*) \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{de Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{de Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

(*) key to go to clausal (Conjunctive Normal Form)
Implication for "humans"; clauses for machines.
de Morgan laws also very useful in going to clausal form.

Style II: Proof by inference rules
Modus Ponens (MP)

KB at T = 1:

R1: $\neg P_{1,1}$

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

How can we show that KR $\models \neg P_{1,2}$?

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

It follows (bicond elim)

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

And-elimination (7.5.1 R&N):

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

By contrapositive:

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

Thus (de Morgan):

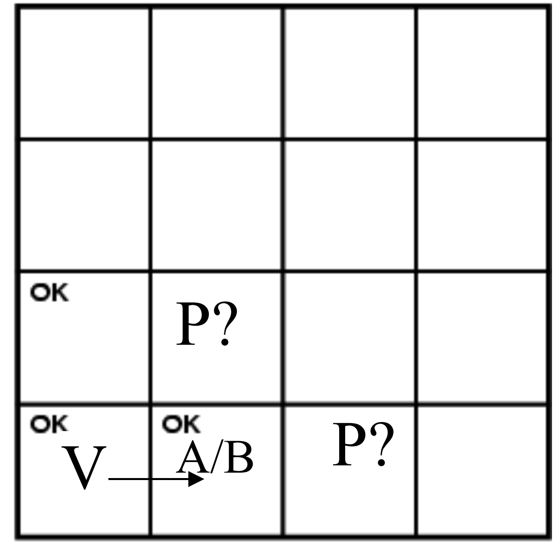
$\neg B_{1,1} \Rightarrow (\neg P_{1,2} \wedge \neg P_{2,1})$

By Modus Ponens using R2, we get

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

Finally, by And-elimination, we get:

$\neg P_{1,2}$



Wumpus world
at T = 1

Note: In formal proof, every step needs to be justified.

Length of Proofs

Why bother with inference rules? We could always use a truth table to check the validity of a conclusion from a set of premises.

But, ***resulting proof can be much shorter than truth table method.***

Consider KB:
$p\_1, \; p\_1 \rightarrow p\_2, \; p\_2 \rightarrow p\_3, \ldots, p\_(n-1) \rightarrow p\_n$

To prove conclusion: p_n

Inference rules: n-1 MP steps   Truth table: 2^n

Key open question: Is there always a short proof for any valid conclusion? Probably not. The NP vs. co-NP question.
(The closely related: P vs. NP question carries a $1M prize.)

First, we need a conversion to Conjunctive Normal Form (CNF) or Clausal Form.

Style III: Resolution

Let's consider converting R4 in clausal form:

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

We have:

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

which gives (implication elimination):

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

Also

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

which gives:

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

Thus,

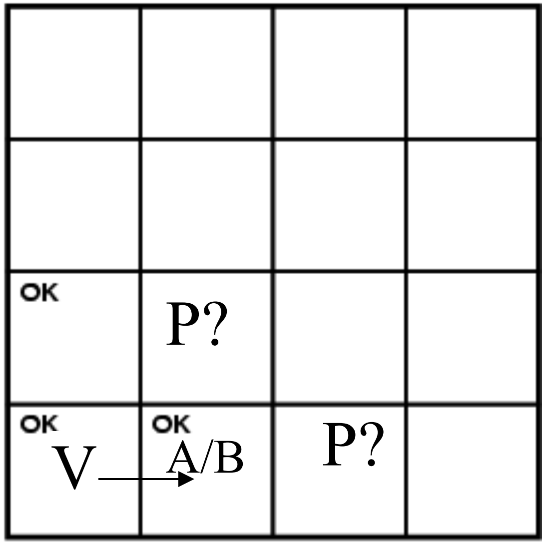
$(\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1}$

leaving,

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

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

(note: clauses in red)



Wumpus world
at T = 1

KB at T = 1:

R1: $\neg P_{1,1}$

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4: $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$

R5: $B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$

KB at T=1 in clausal form:

R1: $\neg P_{1,1}$

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4a: $\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}$

R4b: $\neg P_{1,2} \vee B_{1,1}$

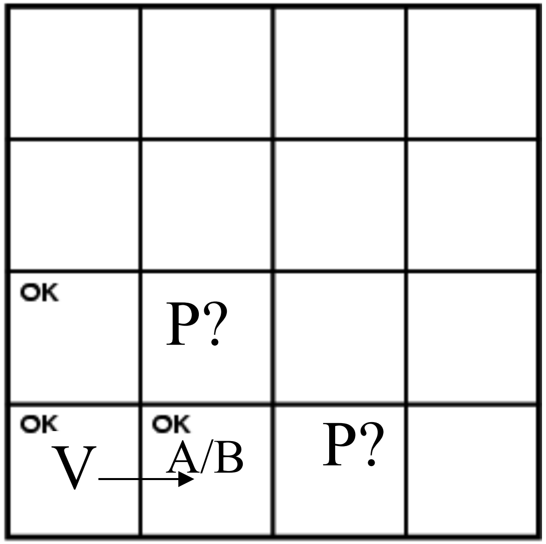
R4c: $\neg P_{2,1} \vee B_{1,1}$

R5a: $\neg B_{2,1} \vee P_{1,1} \vee P_{2,2} \vee P_{3,1}$

R5b: $\neg P_{1,1} \vee B_{2,1}$

R5c: $\neg P_{2,2} \vee B_{2,1}$

R5d: $\neg P_{3,1} \vee B_{2,1}$



Wumpus world at T = 1

How can we show that KR $\models \neg P_{1,2}$?

Proof by contradiction:
Need to show that $(KB \wedge P_{1,2})$ is
inconsistent (unsatisfiable).

Resolution rule:

$(\alpha \vee p)$ and $(\beta \vee \neg p)$

gives resolvent (logically valid conclusion):

$(\alpha \vee \beta)$

If we can reach the empty clause, then KB is inconsistent. (And, vice versa.)

KB at T=1 in clausal form:

R1: $\neg P_{1,1}$

R2: $\neg B_{1,1}$

R3: $B_{2,1}$

R4a: $\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}$

R4b: $\neg P_{1,2} \lor B_{1,1}$

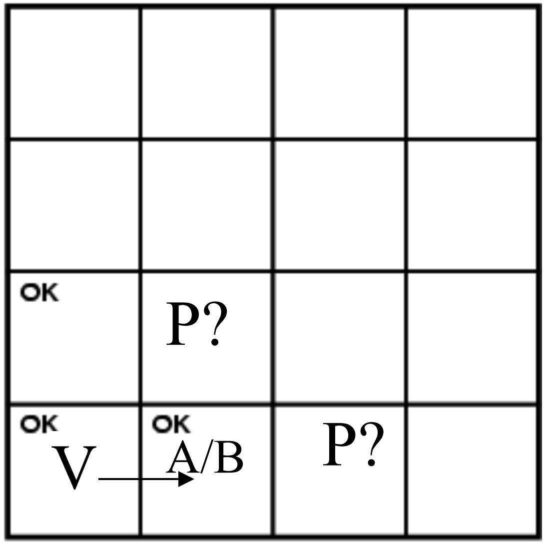
R4c: $\neg P_{2,1} \lor B_{1,1}$

R5a: $\neg B_{2,1} \lor P_{1,1} \lor P_{2,2} \lor P_{3,1}$

R5b: $\neg P_{1,1} \lor B_{2,1}$

R5c: $\neg P_{2,2} \lor B_{2,1}$

R5d: $\neg P_{3,1} \lor B_{2,1}$



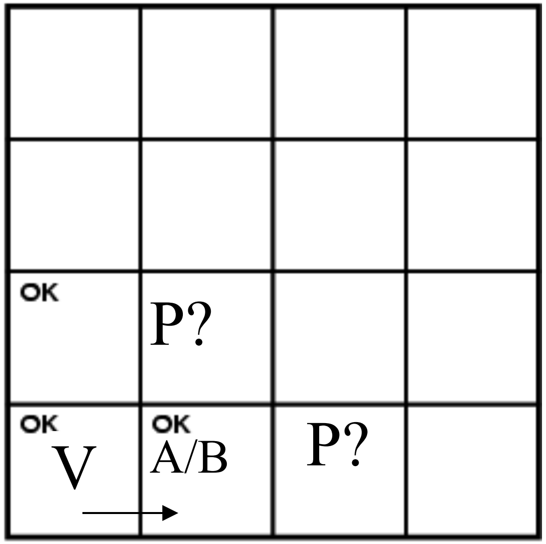
Wumpus world
at T = 1

Show that (KB $\land$ $P_{1,2}$) is **inconsistent. (unsatisfiable)**

R4b with $P_{1,2}$ resolves to $B_{1,1}$,
which with R2, resolves to the empty clause, □ .
So, we can conclude KB $\models \neg P_{1,2}$.
(make sure you use "what you want to prove.")

KB at T=1 in clausal form:

R1: $\neg P_{1,1}$
R2: $\neg B_{1,1}$
R3: $B_{2,1}$
R4a: $\neg B_{1,1} \lor P_{1,2} \lor P_{2,1}$
R4b: $\neg P_{1,2} \lor B_{1,1}$
R4c: $\neg P_{2,1} \lor B_{1,1}$
R5a: $\neg B_{2,1} \lor P_{1,1} \lor P_{2,2} \lor P_{3,1}$
R5b: $\neg P_{1,1} \lor B_{2,1}$
R5c: $\neg P_{2,2} \lor B_{2,1}$
R5d: $\neg P_{3,1} \lor B_{2,1}$



Wumpus world at T = 1

Note that R5a resolved with R1, and then resolved with R3, gives $(P_{2,2} \lor P_{3,1})$.

Almost there… to show KB $\models (P_{2,2} \lor P_{3,1})$, we need to show KB $\land (\neg (P_{2,2} \lor P_{3,1}))$ is inconsistent. (Why? Semantically?)
So, show KB $\land \neg P_{2,2} \land \neg P_{3,1}$ is inconsistent.
This follows from $(P_{2,2} \lor P_{3,1})$; because in two more resolution steps, we get the empty clause (a contradiction).

Length of Proofs

What is hard for resolution?

Consider:

Given a fixed pos. int. N

$(P(i,1) \vee P(i,2) \vee \ldots P(i,N))$ for $i = 1, \ldots N+1$

$(\neg P(i,j) \vee \neg P(i',j))$ for $j = 1, \ldots N$;
$i = 1, \ldots N+1$;
$i' = 1, \ldots N+1$; $i \neq i'$

What does this encode?

Think of: P(i,j) for "object i in location j"

Pigeon hole problem…

Provable requires exponential number of resolution steps to reach empty clause (Haken 1985). Method "can't count."

Style IV: SAT Solvers

Instead of using resolution to show that

$KB \wedge \neg \alpha$ is inconsistent,

modern Satisfiability (SAT) solvers operating on the clausal form are *much* more efficient.

The SAT solvers treat the set of clauses as a set of constraints (disjunctions) on Boolean variables, i.e., a CSP problem!
Current solvers are very powerful. Can handle 1 Million+ variables and several millions of clauses.

Systematic: Davis Putnam (DPLL) + *series of improvements*
Stochastic local search: WalkSAT (issue?)

See R&N 7.6. "Ironically," we are back to semantic model checking, but way more clever than basic truth assignment enumeration (exponentially faster)!

DPLL improvements

Backtracking + …

1) **Component analysis (disjoint sets of constraints? Problem decomposition?)**
2) **Clever variable and value ordering (e.g. degree heuristics)**
3) **Intelligent backtracking and clause learning (conflict learning)**
4) **Random restarts (heavy tails in search spaces…)**
5) **Clever data structures**

1+ Million Boolean vars & 10+ Million clause/constraints are feasible nowadays. (e.g. Minisat solver)

Has changed the world of verification (hardware/software) over the last decade (incl. Turing award for Clarke). Widely used in industry, Intel, Microsoft, IBM etc.

Satisfiability (SAT/CSP) or Inference?

Really solving the same problem but SAT/CSP view appears more effective.

$KB \models \alpha$ iff $M(KB) \subseteq M(\alpha)$ iff $(KB \wedge \neg\alpha)$ is unsat

Assume KB and α is in CNF (clausal form).
Note that: $KB \models (\beta \wedge \gamma)$ iff
$(KB \models \beta)$ and $(KB \models \gamma)$
(consider defn. in terms of models)
So, can break conjunction in several queries, one for each disjunction (clause). Each a call to a SAT solver to check $KB \wedge \neg (l1 \vee l2 \dots)$ for consist. or equivalently check $KB \wedge \neg l1 \wedge \neg l2 \dots$ (i.e. CNF form) for consistency.

KB $\models$ (l1 $\vee$ l2 …) iff KB $\wedge$ $\neg$ l1 $\wedge$ $\neg$ l2 …

The SAT solvers essentially views the KB as a set of constraints that defines a subset of the set of all 2^N possible worlds (N prop. vars.).

The query α also defines a subset of the 2^N possible worlds. (Above we used a single clause (l1 $\vee$ l2 …) where l_i is a var or its negation.)

The SAT/CSP solvers figures out whether there are any worlds consistent with the KB constraints that **do not satisfy the constraints of the query**. If so, than the query does not follow from the KB.

Aside: In verification, such a world exposes a bug.

If such worlds do not exist, then the query must be entailed by the KB. ($M(KB) \subseteq M(\alpha)$ Search starts from query!)

Viewing logical reasoning as reasoning about constraints on the way the "world can be" is quite actually quite natural! It's definitely a computationally effective strategy.

Addendum: Reflections on Inference, Knowledge, and Data ca. 2012

In the logical agent perspective, we take the "knowledge-based" approach.

We'll have a knowledge base, capturing our world knowledge in a formal language. We'll use an inference procedure to derive new information.

Representation has well-defined syntax and semantics / meaning.

How far are actual AI systems in the knowledge / reasoning paradigm?

Specifically, where do Deep Blue, Watson, and Siri fit?

What about IR, Google Translate, and Google's Knowledge Graph?

And, "shallow semantics" / Semantic Web?

IBM's Jeopardy! playing system.

Watson

The basic layout of the *Jeopardy!* game board, using the dollar values from the first round

Chris Welty from IBM on Watson and "understanding"

http://spectrum.ieee.org/podcast/at-work/innovation/what-is-toronto

See 4 min mark for discussion on "understanding."
Till around min 11.

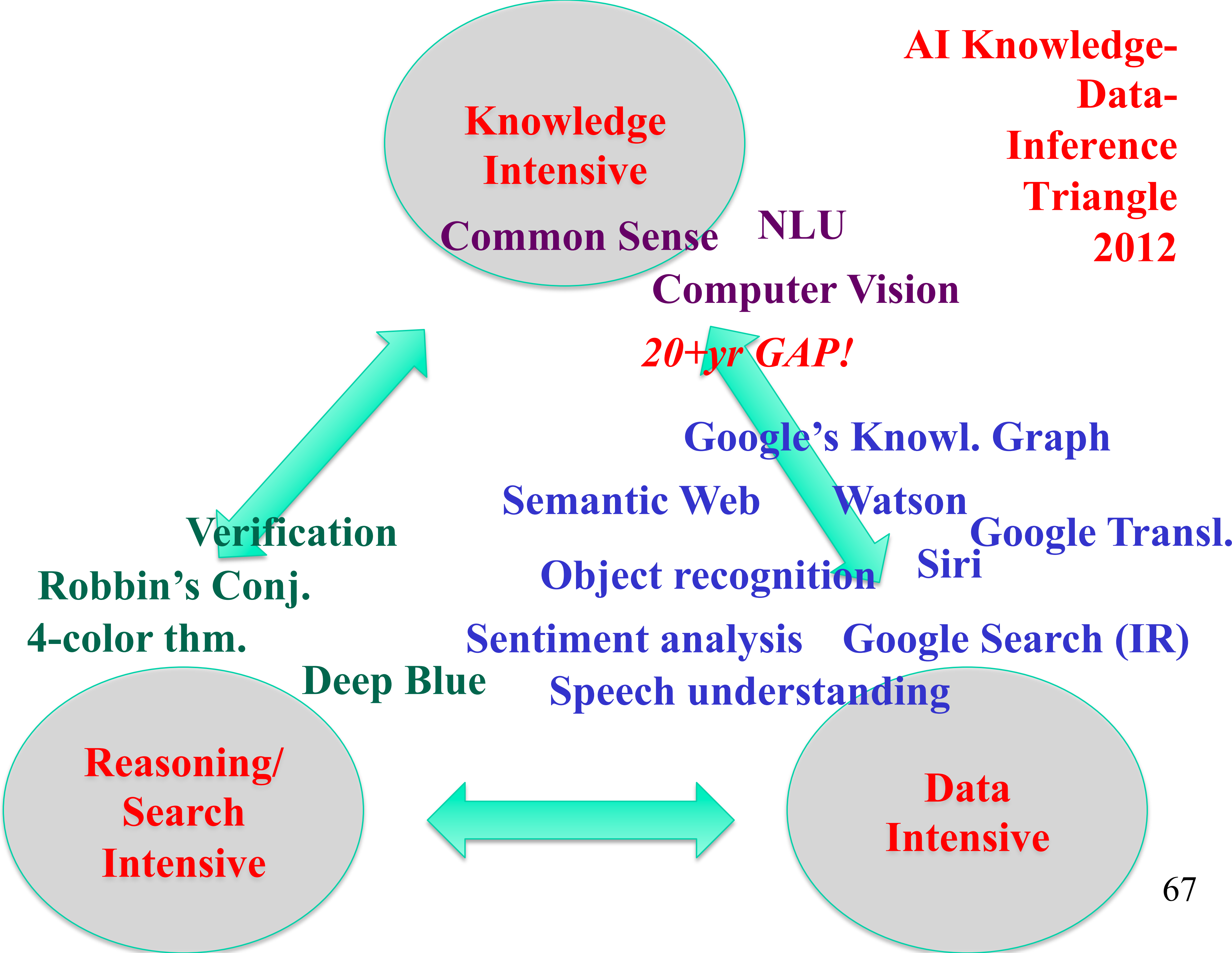
Key aspects:

Watson

1) Limited number of categories of types of questions
2) Well-defined knowledge sources (Wikipedia; Encyclopedias; Dictionaries etc.) Not: WWW (Contrast: Google Knowl. Graph)
3) Around 70 “expert” modules that handle different aspects of the possible question and answers. 1000+ hypotheses
4) Final result based on a confidence vote.
5) Limited language parsing. Lot based on individual words (like IR).
6) Form of IR “+” (parsing; limited knowledge graph: “books have authors” “movie has actors and director” “verb represents an action” etc.)
7) Some game theory to reason about whether to answer

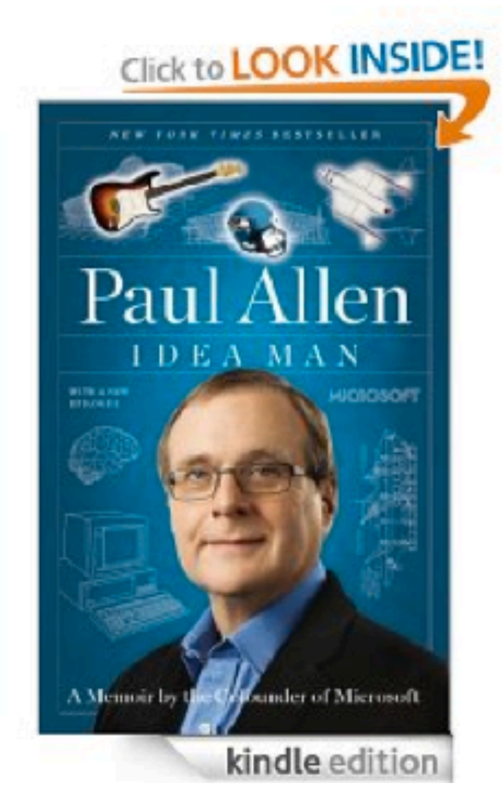
To some extent: Watson “understands”!
(It’s actually a bit of a problem for IBM… Clients expect too much!)

AI Knowledge-Data-Inference Triangle 2012



Interesting readings: Appendix on AI in Paul Allen’s autobiography. Discusses the remaining challenges: mostly focused on commonsense reasoning and knowledge representation.

New: Vulcan Ventures AI Institute.



Idea Man: A Memoir by the Cofounder of Microsoft [Kindle Edition]

Paul Allen (Author)

★★★★☆ (78 customer reviews)

Print List Price: ~~$17.00~~

Kindle Price: $7.99

You Save: $9.01 (53%)

Sold by: Penguin Group (USA) LLC

- Length: 384 pages
- Don't have a Kindle? Get your Kindle here.

Formats	Amazon Price	New from	Used from
Kindle Edition	$7.99	--	--

chemical properties. Indeed, existing artificial intelligence technologies can answer questions that depend only on simple facts. ("How many chromosomes does a blue jay have?"). But the most important elements of human knowledge involve much more sophisticated constructions. Even cut-and-dried knowledge includes rough statements of causality ("Too little sunlight can lead to stunted plants"), generality ("Most birds can fly"), metaphor ("DNA is like a blueprint"), counterfactuals ("If Earth's gravity were halved, trees could be twice as tall"), rule knowledge ("If a cell dies, its cell membrane disintegrates"), and prediction ("Mutations should increase in the presence of radioactivity").

Project Halo: Read biology textbook to be able to pass AP Biology exam. Very challenging! At least, 5 to 10 more years.

Additional Slides
Some examples of SAT solving

I.e. Propositional Reasoning Engines

Application: I --- Diagnosis

- Problem: diagnosis a malfunctioning device
  - Car
  - Computer system
  - Spacecraft
- where
  - Design of the device is known
  - We can observe the state of only <u>certain parts</u> of the device – much is <u>hidden</u>

Model-Based, Consistency-Based Diagnosis

- Idea: create a logical formula that describes how the device should work
  - Associated with each "breakable" component C is a proposition that states "C is okay"
  - Sub-formulas about component C are all conditioned on C being okay
- A diagnosis is a smallest of "not okay" assumptions that are consistent with what is actually observed

Consistency-Based Diagnosis

1. Make some Observations O.
2. Initialize the Assumption Set A to assert that all components are working properly.
3. Check if the KB, A, O together are inconsistent (can deduce *false*).
4. If so, delete propositions from A until consistency is restored (cannot deduce *false*). The deleted propositions are a diagnosis.

There may be many possible diagnoses

Example: Automobile Diagnosis

- *Observable Propositions:*

  EngineRuns, GasInTank, ClockRuns
- *Assumable Propositions:*

  FuelLineOK, BatteryOK, CablesOK, ClockOK
- *Hidden (non-Assumable) Propositions:*

  GasInEngine, PowerToPlugs
- *Device Description F:*

  (GasInTank ∧ FuelLineOK) → GasInEngine

  (GasInEngine ∧ PowerToPlugs) → EngineRuns

  (BatteryOK ∧ CablesOK) → PowerToPlugs

  (BatteryOK ∧ ClockOK) → ClockRuns
- *Observations:*

  ¬ EngineRuns, GasInTank, ClockRuns

Note: of course a highly simplified set of axioms.

Example

(GasInTank ∧ FuelLineOK) → GasInEngine
(GasInEngine ∧ PowerToPlugs) → EngineRuns

- *Is* F ∪ Observations ∪ Assumptions *consistent?*
- F ∪ {¬EngineRuns, GasInTank, ClockRuns} ∪ { FuelLineOK, BatteryOK, CablesOK, ClockOK } → *false*
  - *Must restore consistency!*
- F ∪ {¬EngineRuns, GasInTank, ClockRuns} ∪ { BatteryOK, CablesOK, ClockOK } → *satisfiable*
  - ¬ *FuelLineOK is a diagnosis*
- F ∪ {¬EngineRuns, GasInTank, ClockRuns} ∪ {FuelLineOK, CablesOK, ClockOK } → *false*
  - ¬ *BatteryOK is* *<u>not</u>* *a diagnosis*

Complexity of Diagnosis

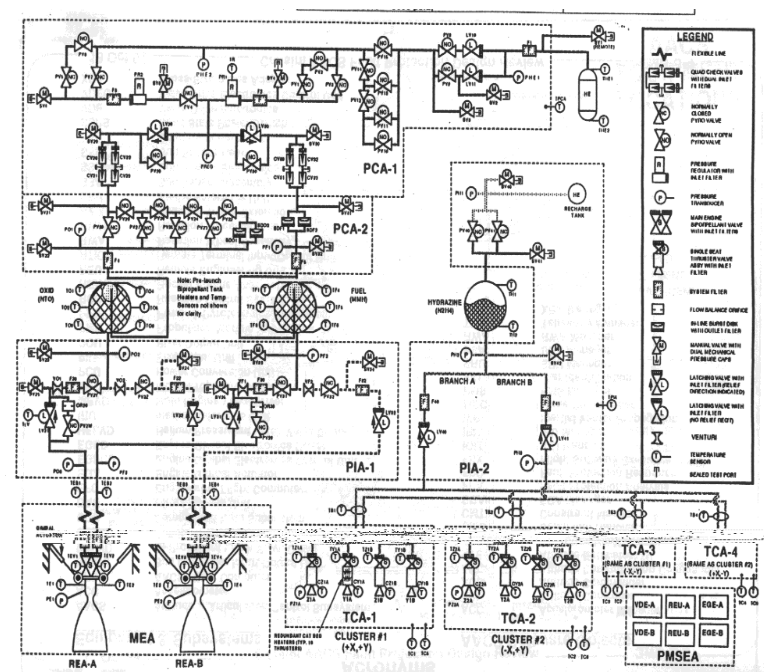
- If F is Horn, then each consistency test takes linear O(n) time – unit propagation is complete for Horn clauses.
- Complexity = ways to delete propositions from Assumption Set that are considered.
  - Single fault diagnosis – $O(n^2)$
  - Double fault $\binom{n}{2}$ diagnosis – $O(n^3)$
  - Triple fault diagnosis – $O(n^4)$

In practice, for non-Horn use SAT solver for consistency check.

Horn clause: at most one positive literal. Also, consider $\Rightarrow$

NASA Deep Space One

- Autonomous diagnosis & repair "Remote Agent"
- Compiled systems schematic to 7,000 var SAT problem



Deep Space One

- a failed electronics unit
  - Remote Agent fixed by reactivating the unit.

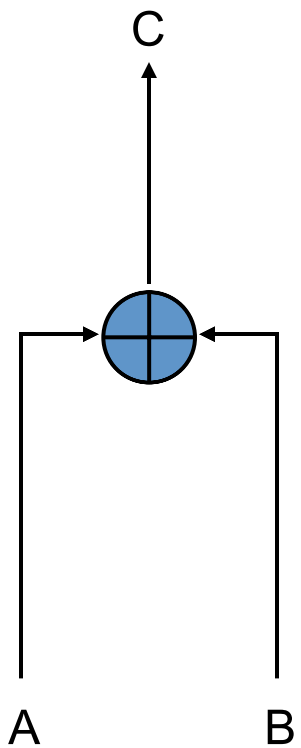
- a failed sensor providing false information
  - Remote Agent recognized as unreliable and therefore correctly ignored.

- an altitude control thruster (a small engine for controlling the spacecraft's orientation) stuck in the "off" position
  - Remote Agent detected and compensated for by switching to a mode that did not rely on that thruster.

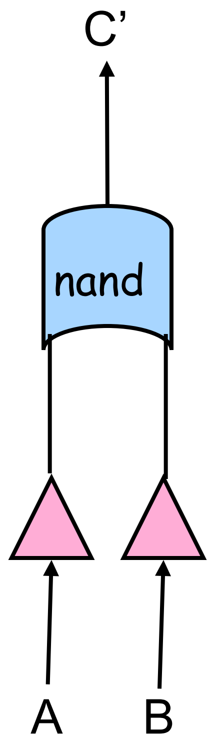
II --- Testing Circuit Equivalence

- Do two circuits compute the same function?
- Circuit optimization
- Is there input for which the two circuits compute different values?

  (Satisfying assignment will tell us. Reveals bug!)

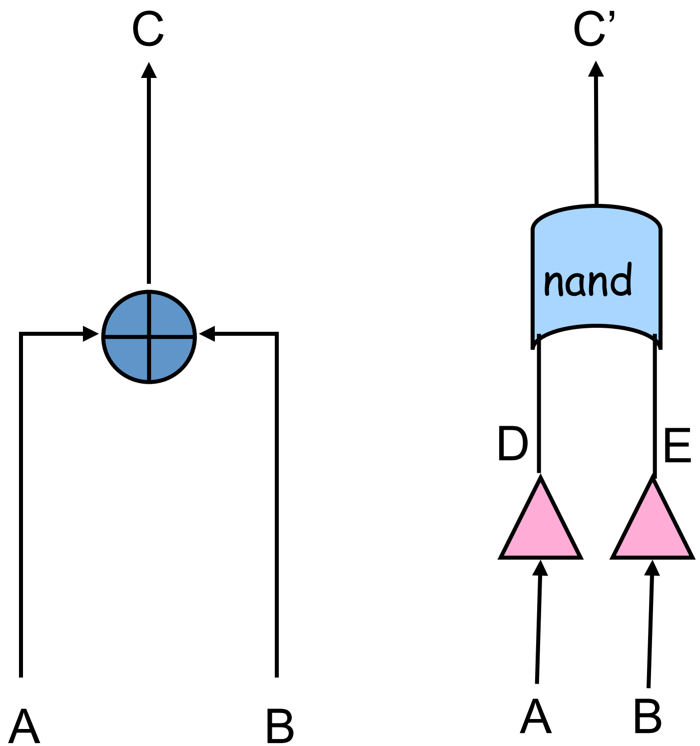


Formal spec. OR



Possible implementation using hardware gates (invertor & nand)

Note: if consitent, model reveals "bug" in hardware design.



$$C \equiv (A \vee B)$$ **Descr. of OR**

$$C' \equiv \neg(D \wedge E)$$
$$D \equiv \neg A$$
$$E \equiv \neg B$$

Descr. of Hardware.

$$\neg(C \equiv C')$$

Our query: $(C \equiv C')$
Does this hold?
Yes iff negation with KB is inconsistent.

What happens if you add $(C \equiv C')$ **instead of** $\neg(C \equiv C')$**? Still OK?**

Informally: Given the same inputs values for A and B, logic specifies outputs (C and C'). Are there inputs for which C and C' differ?

III --- SAT Translation of N-Queens

- At least one queen each row:

  (Q11 v Q12 v Q13 v ... v Q18)

  (Q21 v Q22 v Q23 v ... v Q28)

  ...

- No attacks (columns):

  (~Q11 v ~Q21)

  (~Q11 v ~Q31)

  (~Q11 v ~Q41)

  ...

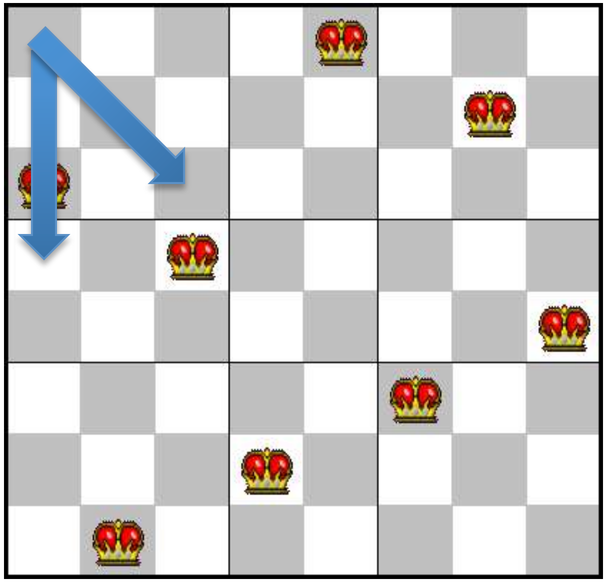
- No attacks (diag; need left and right):

  (~Q11 v ~Q22)

  (~Q11 v ~Q33)

  (~Q11 v ~Q44)

  ...



How about: No attacks (rows)

(~Q11 v ~Q12)

(~Q11 v ~Q13)

(~Q11 v ~Q14)

...

Redundant! Why? Sometimes slows solver.

More compactly

- At least one queen each row:
  (Qi1 v Qi2 ... v QiN) for 1 <= i <= N
- No attacks (columns; "look in i^{th} column"):
  (~Qji v ~Qj'i) for 1 <= i,j,j' <= N and j =/= j'
- No attacks (diag; need left and right):
  (~Qij v ~Qi'j') for 1 <= i, i', j, j' s.t. |i – i'| = |j – j'| & i =/= i' & j =/= j'

Or: in first order logic syntax, e.g., second constraint set:

$\forall$ i $\forall$ j $\forall$ j' (j =/= j' $\Rightarrow$ ($\neg$ $Q_{j,i}$ $\vee$ $\neg$ $Q_{j',i}$))

with bounded type quantifier 1 <= i, j, j' <= N

Really a compact "propositional schema."

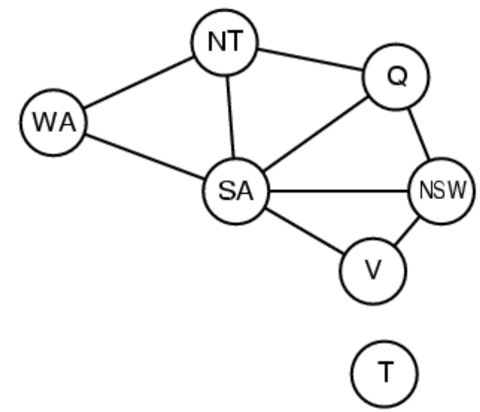
First-order logic for finite domains is equiv. to prop. logic.

For SAT solver, always "ground to propositional."

IV --- SAT Translation of Graph Coloring

At least one color per node i :

(C_i1 v Q_i2 v Q_i3 v ... v Q_iK)



At most one color per node i :

(~C_ik v ~Q_ik') for all k =/= k'

If node i and node j (=/= i) share an edge, need to have different colors:

(~C_il v ~Q_jl) for all 1 <= l <= K

Note: Translation from "problem" into SAT. Reverse of usual translation to show NP-completeness.

C_ik for node i has color k
Total # colors: K. Total # nodes: N.

Works also for (easy) polytime problems!

V --- Symbolic Model Checking

- Any finite state machine is characterized by a transition function
  - CPU
  - Networking protocol
- Wish to prove some invariant holds for any possible inputs
- Bounded model checking: formula is sat *iff* invariant fails *k* steps in the future

$\overline{S_t}$ = vector of Booleans representing state of machine at time t

$$\rho : State \times Input \rightarrow State$$

$$\gamma : State \rightarrow \{0,1\}$$

$$\left(\bigwedge_{i=0}^{k-1} \left(\overline{S_{i+1}} \equiv \rho(\overline{S_i}, \overline{I_i}) \right) \right) \wedge S_o \wedge \neg \gamma(S_k)$$

1) Can go to CNF.
2) Equivalent to planning as propositional inference
SATPLAN
(Kautz & Selman 1996)
7.7.4 R&N
The k-step plan is satisfying assignment.

Note: ρ is just like our "move" earlier. Axioms says what happens "next."

A real-world example

From "SATLIB":

http://www.satlib.org/benchm.html

SAT-encoded bounded model checking instances (contributed by Ofer Shtrichman)

In Bounded Model Checking (BMC) [BCCZ99], a rather newly introduced problem in formal methods, the task is to check whether a given model M (typically a hardware design) satisfies a temporal property P in all paths with length less or equal to some bound k. The BMC problem can be efficiently reduced to a propositional satisfiability problem, and in fact if the property is in the form of an invariant (Invariants are the most common type of properties, and many other temporal properties can be reduced to their form. It has the form of 'it is always true that ... '.), it has a structure which is similar to many AI planning problems.

Bounded Model Checking instance

The instance bmc-ibm-6.cnf, IBM LSU 1997:

```
p cnf 51639 368352
-1 7 0
-1 6 0
-1 5 0
-1 -4 0
-1 3 0
-1 2 0
-1 -8 0
-9 15 0
-9 14 0
-9 13 0
-9 -12 0
-9 11 0
-9 10 0
-9 -16 0
-17 23 0
-17 22 0
```

i.e. **((not x_1) or x_7)**
and ((not x_1) or x_6)
and … *etc.*

10 pages later:

185 −9 0
185 −1 0
177 169 161 153 145 137 129 121 113 105 97
89 81 73 65 57 49 41
33 25 17 9 1 −185 0
186 −187 0
186 −188 0

…

(x_{177} or x_{169} or x_{161} or x_{153} …
or x_{17} or x_9 or x_1 or (not x_{185}))

clauses / constraints are getting more interesting…

4000 pages later:

10236 −10050 0
10236 −10051 0
10236 −10235 0
10008 10009 10010 10011 10012 10013 10014
10015 10016 10017 10018 10019 10020 10021
10022 10023 10024 10025 10026 10027 10028
10029 10030 10031 10032 10033 10034 10035
10036 10037 10086 10087 10088 10089 10090
10098 10099 10100 10101 10102 10103 10104
10105 10106 10107 10108 −55 −54 53 −52 −51 50
10047 10048 10049 10050 10051 10235 −10236 0
10237 −10008 0
10237 −10009 0
10237 −10010 0

...

!!!

a 59-cnf clause…

Finally, 15,000 pages later:

$-7\ 260\ 0$
$7\ -260\ 0$
$1072\ 1070\ 0$
$-15\ -14\ -13\ -12\ -11\ -10\ 0$
$-15\ -14\ -13\ -12\ -11\ 10\ 0$
$-15\ -14\ -13\ -12\ 11\ -10\ 0$
$-15\ -14\ -13\ -12\ 11\ 10\ 0$
$-7\ -6\ -5\ -4\ -3\ -2\ 0$
$-7\ -6\ -5\ -4\ -3\ 2\ 0$
$-7\ -6\ -5\ -4\ 3\ -2\ 0$
$-7\ -6\ -5\ -4\ 3\ 2\ 0$
$185\ 0$

What makes this possible?

Note that: $2^{50000} \approx 3.160699437 \cdot 10^{15051}$ ***... !!!***

The Chaff SAT solver (Princeton) solves this instance in less than one minute.

Progress in Last 20 years

- *Significant progress since the 1990's.* How much?
- Problem size: **We went from 100 variables, 200 constraints (early 90's) to 1,000,000+ variables and 5,000,000+ constraints in 20 years**

- **Search space: from 10^30 to 10^300,000.**
  [Aside: "one can encode quite a bit in 1M variables."]

- **Is this just Moore's Law? It helped, but not much…**
  - **2x faster computers does *not* mean can solve 2x larger instances**
  - **search difficulty does *not* scale linearly with problem size!**

    In fact, for O(2^n), 2x faster, how many more vars?

    handles 1 more variable!!

  Mainly algorithmic progress. Memory growth also key.

- **Tools: 50+ competitive SAT solvers available (e.g. Minisat solver)**
- **See http://www.satcompetition.org/**

Forces Driving Faster, Better SAT Solvers
Inference engines

- **From academically interesting to practically relevant "Real" benchmarks**, with real interest in solving them

- Regular **SAT Solver Competitions** (Germany-89, Dimacs-93, China-96, SAT-02, SAT-03, …, SAT-07, SAT-09, SAT-2011)
  - "Industrial-instances-only" **SAT Races** (2008, 2010)
  - A tremendous resource! E.g., SAT Competition 2006 (Seattle):
    - 35+ solvers submitted, downloadable, mostly open source
    - 500+ industrial benchmarks, 1000+ other benchmarks
    - 50,000+ benchmark instances available on the Internet

- ***Constant improvement in SAT solvers is the key to the success of, e.g., SAT-based planning, verification, and KB inference.***