Logical Agents

Example: The Wumpus-World Game

Example: The Wumpus-World Game

The **wumpus world** is a cave consisting of rooms connected by passageways.

Lurking somewhere in the cave is the terrible wumpus, a beast that eats anyone who enters its room.

The wumpus can be shot by an agent, but the agent has only one arrow.

Some rooms contain bottomless pits that will trap anyone who wanders into these rooms (except for the wumpus, which is too big to fall in).

The only mitigating feature of this bleak environment is the possibility of finding a heap of gold.

Wumpus-World Game

Performance measure

gold +1000, death -1000

-1 per step, -10 for using the arrow

Environment

Squares adjacent to wumpus are smelly

Squares adjacent to a 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

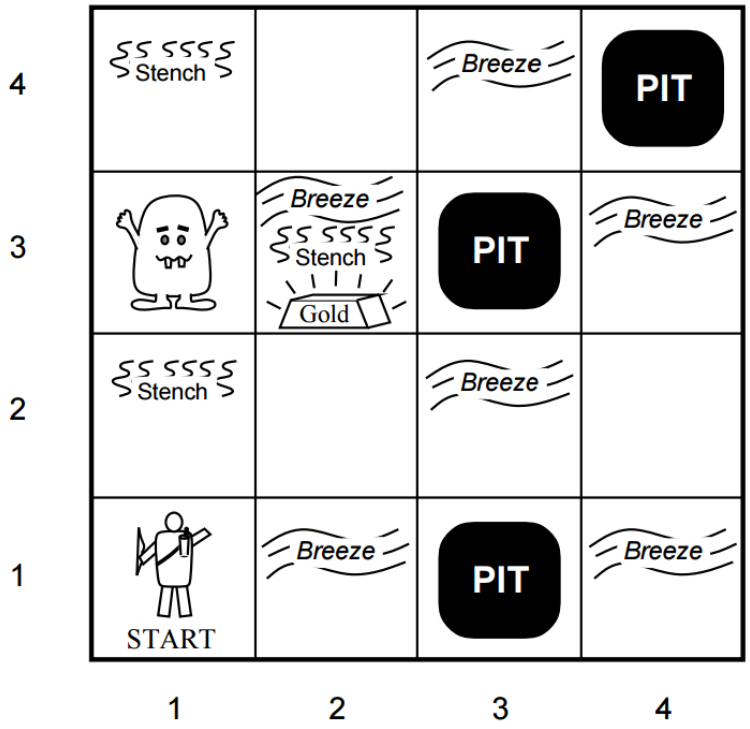
Actuators

Left turn, Right turn,

Forward, Grab, Release, Shoot

Sensors

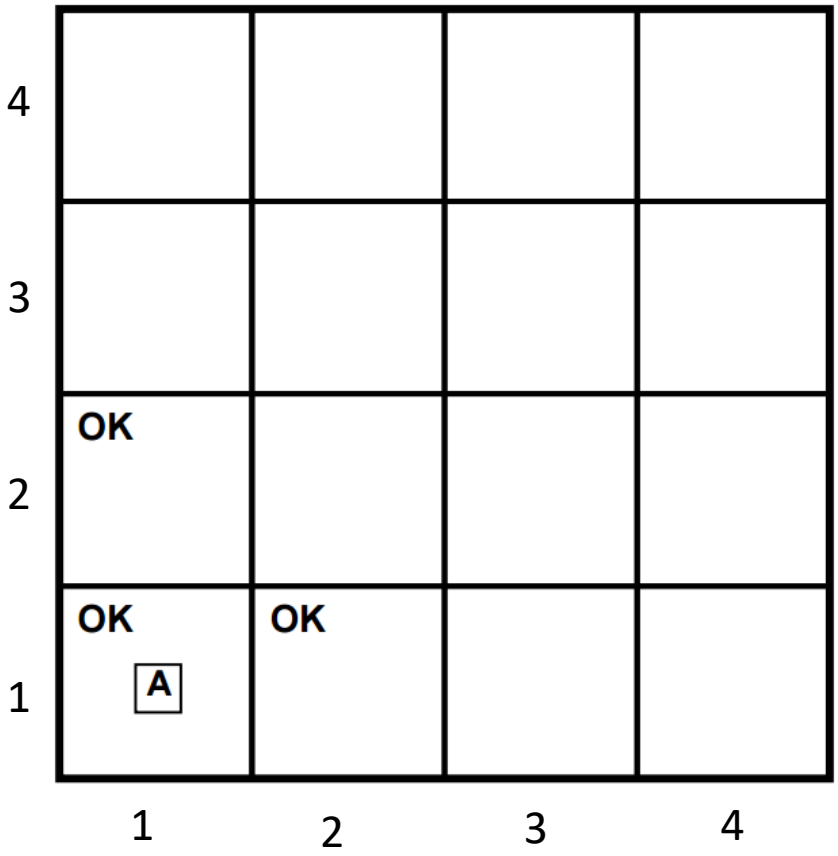
Breeze, Glitter, Smell



Pits, wumpus, and gold are randomly positioned

Exploring a wumpus world

For this example, we use an **informal knowledge representation language** consisting of writing down symbols in a grid.

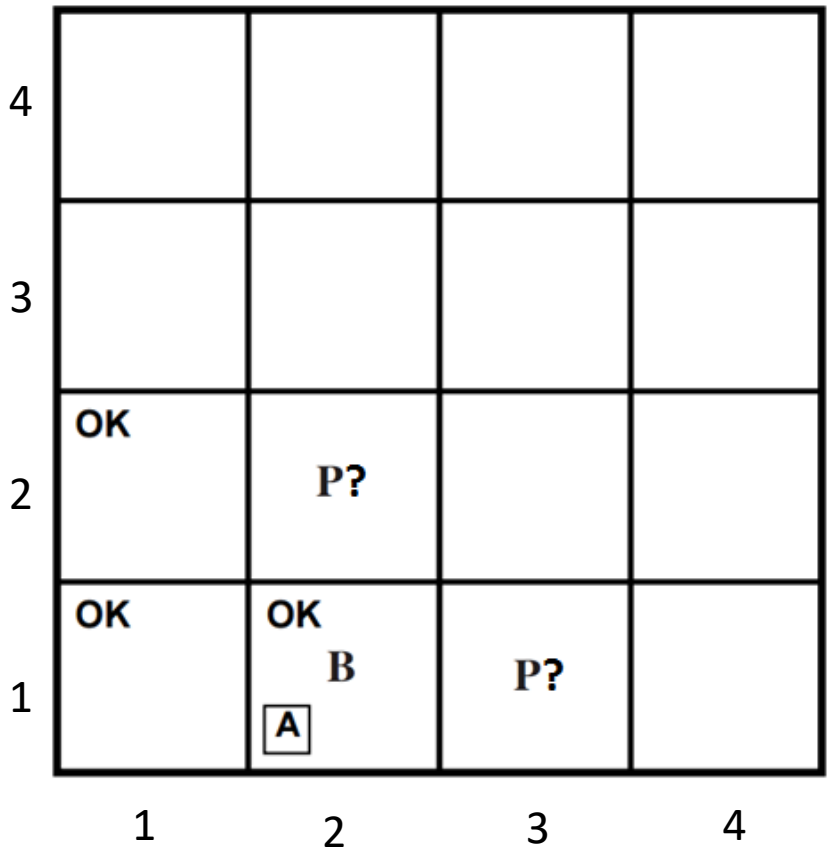


A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Agent knows it is in [1,1] and that the square is **safe**;

The first percepts are all **None**, so, the agent concludes its neighboring squares are free of dangers.

Exploring a wumpus world



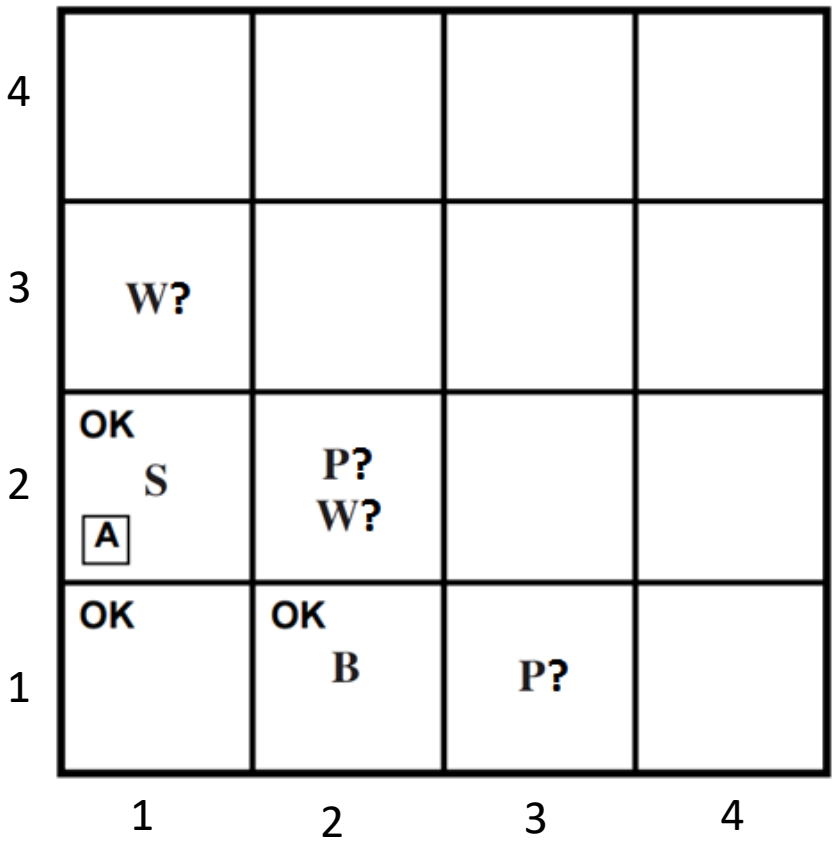
A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

Agent decides to move to [2,1] and there it feels a **breeze**.

At this point, there is only one known square that is OK and that has not yet been visited, [1,2].

So, the prudent agent will turn around, go back to [1,1], and then proceed to [1,2].

Exploring a wumpus world



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

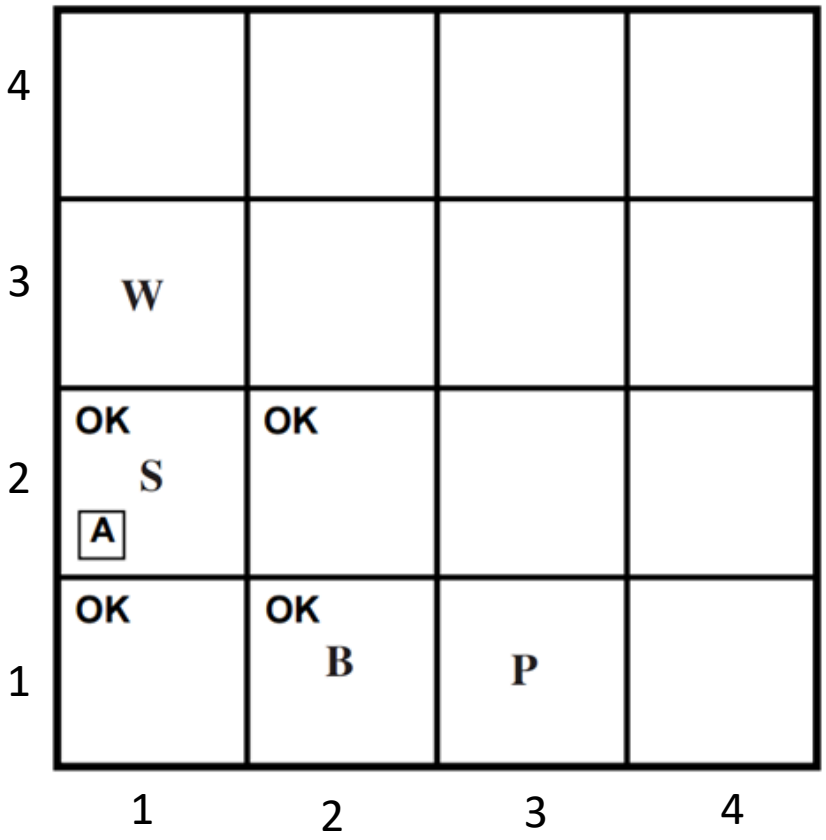
The agent smells stench in [1,2]. So, there is a wumpus in [1,3] or [2,2].

Can we really have a pit or a wumpus in [2,2]?

Exploring a wumpus world

Wumpus can't be in [2,2] or the agent would have detected a stench when it was in [2,1].

Therefore, the agent infers wumpus is in [1,3].



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

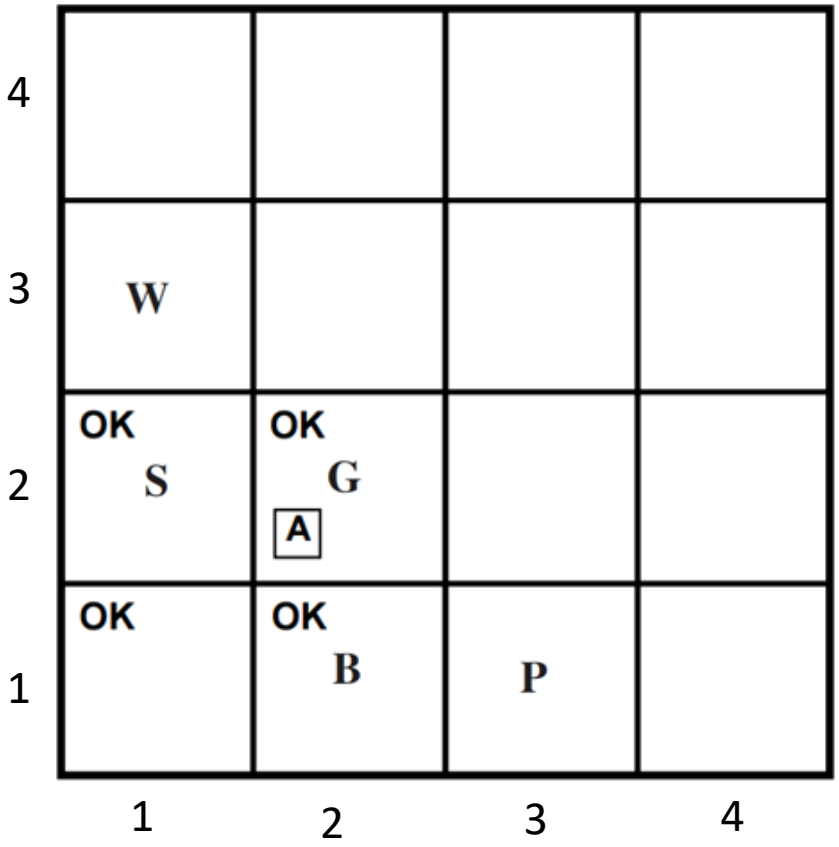
Moreover, the lack of breeze in [1,2] implies there is no pit in [2,2].

Yet the agent has already inferred that there must be a pit in either [2,2] or [3,1],

so this means the pit must be in [3,1].

Fairly difficult inference!

Exploring a wumpus world



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

Agent moves to [2,2] and perceives glitter.

Agent grabs the gold and goes home!

Logic in general

- **Logics** are formal languages for representing information
  - such that conclusions can be drawn
- **Syntax** defines the sentences
  - allowed in the language
- **Semantics** define the "meaning" of sentences;
  - i.e., define the truth of sentences in a world

Propositional Logic

SYNTAX

Propositional Constants

- Examples:
  - *raining*
  - *r*32*aining*
  - *rAiNiNg*
  - *raining_or_snowing*
- Non-Examples:
  - 324567
  - *raining.or.snowing*

Compound Sentences

- Negations:

  $\neg$*raining*

- Conjunctions:

  raining $\wedge$ *snowing*

- Disjunctions:

  raining $\vee$ *snowing*

Compound Sentences

- Implications:

  raining $\Rightarrow$ *cloudy*

  - The left argument of an implication is the *antecedent*.
  - The right argument of an implication is the *consequent*.

- Reductions:

  cloudy $\Leftarrow$ *raining*

  - The left argument of a reduction is the *consequent*.
  - The right argument of a reduction is the *antecedent*.

- Equivalences:

  cloudy $\Leftrightarrow$ *raining*

Operator Precedence

(helps reduce parentheses used)

$\neg$

$\wedge$

$\vee$

$\Leftarrow \Leftrightarrow \Rightarrow$

E.g. $q \vee s \Rightarrow r$ no need for parentheses here

Parenthesis still needed though

E.g.

$(p \Rightarrow q) \vee (s \Rightarrow r)$

Wumpus Sentences

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

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

$\neg P_{1,1}$

$\neg B_{1,1}$

$B_{2,1}$

"Pits cause breezes in adjacent squares"

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

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

Different way to say it:

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

SEMANTICS

Interpretation

- An *interpretation (**i**)* is an association between the propositional **constants** and **truth values** T or F.

$p \xrightarrow{i} \mathrm{T}$

$q \xrightarrow{i} \mathrm{F}$

$r \xrightarrow{i} \mathrm{T}$

written also

$p^i = \mathrm{T}$

$q^i = \mathrm{F}$

$r^i = \mathrm{T}$

Operator Semantics

From a given *interpretation* we can derive the truth of any sentence based on the recursive application of the following rules.

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

Evaluation

- Consider the interpretation *i* shown below.

  $p^i = true$

  $q^i = false$

  $r^i = true$

- We can see that *i* satisfies $(p \vee q) \wedge (\neg q \vee r)$

$$(true \vee false) \wedge (\neg false \vee true)$$

$$true \wedge (\neg false \vee true)$$

$$true \wedge (true \vee true)$$

$$true \wedge true$$

$$true$$

Evaluation (continued)

- Now consider interpretation *j* shown below.

  $p^j = true$

  $q^j = true$

  $r^j = false$

- In this case, *j* does not satisfy $(p \vee q) \wedge (\neg q \vee r)$

$$(true \vee true) \wedge (\neg true \vee false)$$

$$true \wedge (\neg true \vee false)$$

$$true \wedge (false \vee false)$$

$$true \wedge false$$

$$false$$

Satisfaction and Models

- If based on a given interpretation a sentence is determined to be true (we say “satisfied”), the interpretation is called a model of that sentence.
  - We also say “the sentence is true in that model”

Models and entailment

Let α be a sentence.

We say m is a model of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

α **entails** β, denoted by $\alpha \models \beta$ if and only if $M(\alpha) \subseteq M(\beta)$

E.g.

Sentence p entails sentence $p \vee q$.

Since a disjunction is true whenever (in every model that) one of its disjuncts is true, then $p \vee q$ must be true whenever p is true.

On the other hand, sentence p does *not* logically entail $p \wedge q$.

A conjunction is true if and only if *both* of its conjuncts are true, and q may be false.

Knowledge Bases

A KB (knowledge base) is a set of sentences.

We say m is a model of KB **if** all sentences of KB are true in m.

Validity, Satisfiability, Unsatisfiability

- A sentence is *valid* if and only if it is satisfied by *every* interpretation.

  e.g. $p \vee \neg p$ is valid.

- A sentence is *satisfiable* if and only if it is satisfied by at least one interpretation.

  e.g. most typical sentences

- A sentence is *unsatisfiable* if and only if it is not satisfied by any interpretation.

  e.g. $p \Leftrightarrow \neg p$ is unsatisfiable.

  No matter what interpretation we take, the sentence is always false.

Several basic methods for determining whether a given set of premises propositionally entails a given conclusion.

SEMANTIC REASONING METHODS

Truth-Table Method

One way of determining whether or not a set of sentences (premises) entail a possible conclusion (another sentence) is to check the **truth table**.

Method (Build two truth tables)

1. Start with a complete truth table for the propositional constants.

2. Iterate through all the premises,

   for each premise eliminate any row that does not satisfy the premise.

3. Build a similar table for the conclusion sentence.

4. Finally, compare the two tables.

   If every row that remains in the premise table also remains in the conclusion table, then the premises logically entail the conclusion.

Amy's …

Simple sentences (just propositional constants):

Amy loves Pat.

lovesAmyPat.

Amy loves Quincy.

lovesAmyQuincy

It is Monday.

ismonday

Premises:

If Amy loves Pat, Amy loves Quincy.

$lovesAmyPat \Rightarrow lovesAmyQuincy$

If it Monday, Amy loves Pat or Quincy.

$ismonday \Rightarrow lovesAmyPat \vee lovesAmyQuincy$

Question:

If it is Monday, does Amy love Quincy?

i.e. is $ismonday \Rightarrow lovesAmyQuincy$ entailed by the premises?

Truth table for the premises

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*lovesAmyPat* $\Rightarrow$ *lovesAmyQuincy*	*ismonday* $\Rightarrow$ *lovesAmyPat* $\vee$ *lovesAmyQuincy*
T	*T*	*T*	T	T
T	*T*	*F*	T	T
T	*F*	*T*	F	T
T	*F*	*F*	F	T
F	*T*	*T*	T	T
F	*T*	*F*	T	T
F	*F*	*T*	T	F
F	*F*	*F*	T	T

Crossing out non-sat interpretations

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*lovesAmyPat* ⇒ *lovesAmyQuincy*	*ismonday* ⇒ *lovesAmyPat* ∨ *lovesAmyQuincy*
T	*T*	*T*	T	T
T	*T*	*F*	T	T
T	*F*	*T*	F	T
T	*F*	*F*	F	T
F	*T*	*T*	T	T
F	*T*	*F*	T	T
F	*F*	*T*	T	F
F	*F*	*F*	T	T

Models of premises (each row is a model)

Truth table for the conclusion

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*ismonday* $\Rightarrow$ *lovesAmyQuincy*
T	*T*	*T*	T
T	*T*	*F*	T
T	*F*	*T*	F
T	*F*	*F*	T
F	*T*	*T*	T
F	*T*	*F*	T
F	*F*	*T*	F
F	*F*	*F*	T

Crossing out non-sat interpretations

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*ismonday* ⇒ *lovesAmyQuincy*
T	*T*	*T*	T
T	*T*	*F*	T
T	*F*	*T*	F
T	*F*	*F*	T
F	*T*	*T*	T
F	*T*	*F*	T
F	*F*	*T*	F
F	*F*	*F*	T

Models of conclusion (each row is a model)

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*lovesAmyPat* ⇒ *lovesAmyQuincy*	*ismonday* ⇒ *lovesAmyPat* ∨ *lovesAmyQuincy*
T	*T*	*T*	T	T
T	*T*	*F*	T	T
T	*F*	*T*	F	T
T	*F*	*F*	F	T
F	*T*	*T*	T	T
F	*T*	*F*	T	T
F	*F*	*T*	T	F
F	*F*	*F*	T	T

lovesAmyPat	*lovesAmyQuincy*	*ismonday*	*ismonday* ⇒ *lovesAmyQuincy*
T	*T*	*T*	T
T	*T*	*F*	T
T	*F*	*T*	F
T	*F*	*F*	T
F	*T*	*T*	T
F	*T*	*F*	T
F	*F*	*T*	F
F	*F*	*F*	T

The remaining rows of the first table are a subset of the remaining rows of the second table. Theorem proved. Amy loves Quincy!

Why comparing tables makes sense?

Because it amounts to checking that

Models of premises $\subseteq$ ***Models*** of conclusion

which is the condition for entailment.

Remember: Each row is a model.

Validity checking

Single table approach.

To determine whether a set of sentences

$\{\varphi_1,\ldots,\varphi_n\}$

logically entails a sentence φ, we form the sentence

$(\varphi_1 \wedge \ldots \wedge \varphi_n \Rightarrow \varphi)$

and check that it is **valid**.

E.g. we write

(lovesAmyPat $\Rightarrow$ lovesAmyQuincy) $\wedge$ (ismonday $\Rightarrow$ lovesAmyPat $\vee$ lovesAmyQuincy)
$\Rightarrow$ (ismonday $\Rightarrow$ lovesAmyQuincy)

Then form a truth table with an added column for this sentence and check its satisfaction under each of the possible interpretations for our logical constants.

Unsatisfiability Checking

Another single table approach.

Works negatively instead of positively.

> To determine whether a finite set of sentences $\{\varphi_1,\ldots,\varphi_n\}$ entails a sentence φ, we form the sentence
>
> $$(\varphi_1 \wedge \ldots \wedge \varphi_n \wedge \neg\varphi)$$
>
> and check that it is **unsatisfiable**.

- **Remark.** Both the validity checking method and the satisfiability checking method require about the same amount of work as the truth table method, but they have the merit of manipulating **only one table**.

PROOFS

Why not be happy with semantic methods?

- Semantic methods for checking logical entailment have the merit of being **conceptually simple**; they directly manipulate interpretations of sentences.

- Unfortunately, the number of interpretations of a language grows **exponentially** with the number of propositional constants.

- **Proof methods** provide an alternative way of **checking** and **communicating** logical entailment that addresses this problem.

Rules of Inference

A *rule of inference* is a **pattern of reasoning** consisting of:

a first set of sentence variables, called *premises*, and

a second set of sentence variables, called *conclusions*.

A rule of inference is *sound*

iff

for every instantiation of sentence variables,

premises logically entail conclusions.

E.g. of a sound rule: Modus Ponens

$\varphi \Rightarrow \psi$

φ

ψ

$p \Rightarrow (q \Rightarrow r)$

p

$q \Rightarrow r$

We can substitute for the sentence variables, φ and ψ, complex sentences.

raining $\Rightarrow$ *wet*

raining

---

wet

$(p \Rightarrow q) \Rightarrow r$

$p \Rightarrow q$

---

r

wet $\Rightarrow$ *slippery*

wet

---

slippery

Note that, by stringing together applications of rules of inference, it is possible to derive conclusions that cannot be derived in a single step. This idea of **stringing together rule applications** leads to the **notion of a proof**.

Resolution

- *Resolution* is an extremely powerful **rule of inference**.
- Using resolution, it is possible to build a theorem prover that is **sound** and **complete** for all of Propositional Logic.
- **Soundness Theorem:** If φ is provable (using resolution) from Δ, then Δ logically entails φ.
- **Completeness Theorem:** If Δ logically entails φ, then φ is provable (using resolution) from Δ.

Clausal Form

Resolution works only on expressions in *clausal form*.

Premises and conclusions must be converted to this form.

How?

Some definitions first…

- A *literal* is either an atomic sentence or a negation of an atomic sentence.
  - For example, if p is a logical constant, the following sentences are both literals: p, $\neg p$

- A *clause expression* is either a literal or a disjunction of literals.
  - If p and q are logical constants, then the following are clause expressions p, $\neg p$, $p \vee q$

- A *clause* is the set of literals in a clause expression.
  - For example, the following sets are the corresponding clauses: $\{p\}$, $\{\neg p\}$, $\{p,q\}$
  - Note that the empty set {} is also a clause. It is equivalent to an empty disjunction.

Converting to clausal form (INDO)

1. Implications (I):

$\varphi_1 \Rightarrow \varphi_2 \quad \rightarrow \quad \neg\varphi_1 \vee \varphi_2$

$\varphi_1 \Leftarrow \varphi_2 \quad \rightarrow \quad \varphi_1 \vee \neg\varphi_2$

$\varphi_1 \Leftrightarrow \varphi_2 \quad \rightarrow \quad (\neg\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \neg\varphi_2)$

2. Negations (N):

$\neg\neg\varphi \quad \rightarrow \quad \varphi$

$\neg(\varphi_1 \wedge \varphi_2) \quad \rightarrow \quad \neg\varphi_1 \vee \neg\varphi_2$

$\neg(\varphi_1 \vee \varphi_2) \quad \rightarrow \quad \neg\varphi_1 \wedge \neg\varphi_2$

3. Distribution (D):

$\varphi_1 \vee (\varphi_2 \wedge \varphi_3) \quad \rightarrow \quad (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$

$(\varphi_1 \wedge \varphi_2) \vee \varphi_3 \quad \rightarrow \quad (\varphi_1 \vee \varphi_3) \wedge (\varphi_2 \vee \varphi_3)$

4. Operators (O):

$\varphi_1 \vee \ldots \vee \varphi_n \quad \rightarrow \quad \{\varphi_1, \ldots, \varphi_n\}$

$\varphi_1 \wedge \ldots \wedge \varphi_n \quad \rightarrow \quad \{\varphi_1\} \ldots \{\varphi_n\}$

Examples

$g \wedge (r \Rightarrow f)$

I $g \wedge (\neg r \vee f)$

N $g \wedge (\neg r \vee f)$

D $g \wedge (\neg r \vee f)$

O $\{g\}$

$\{\neg r, f\}$

$\neg(g \wedge (r \Rightarrow f))$

I $\neg(g \wedge (\neg r \vee f))$

N $\neg g \vee \neg(\neg r \vee f)$

$\neg g \vee (\neg\neg r \wedge \neg f)$

$\neg g \vee (r \wedge \neg f)$

D $(\neg g \vee r) \wedge (\neg g \vee \neg f)$

O $\{\neg g, r\}$

$\{\neg g, \neg f\}$

Propositional Resolution

$\{\varphi_1,\ldots, \chi,\ldots,\varphi_m\}$

$\{\psi_1,\ldots,\neg\chi,\ldots,\psi_n\}$

$\{\varphi_1,\ldots,\varphi_m,\psi_1,\ldots,\psi_n\}$

E.g.

$\{p,q\}$

$\{\neg p,r\}$

$\{q,r\}$

The idea of resolution is simple (see example).

- If *p* is false, then by the first clause *q* must be true.
- If *p* is true, then, by the second clause, *r* must be true.
- Since *p* must be either true or false, then it must be the case that *q* is true or *r* is true.
- In other words, we can cancel the *p* literals.

Resolution Process

To determine **logical entailment**, all we need to do is:

negate the goal,

add it to our premises, and

use Resolution

to determine whether the resulting set is **unsatisfiable** (i.e. derive the empty clause).

Reductio ad absurdum

Example

1. $\{p,q\}$ Premise
2. $\{\neg p, q\}$ Premise
3. $\{p,\neg q\}$ Premise
4. $\{\neg p,\neg q\}$ Negated conclusion (will just say “premise” in other examples)
5. $\{q\}$ 1,2
6. $\{\neg q\}$ 3,4
7. $\{\}$ 5,6

Two finger (or pointer) method

- We keep two pointers (the ***slow*** and the ***fast***);
- On each step we compare the sentences under the pointers. If we can resolve, we add the new derived sentence at the end of the list.
- At the start of the inference we initialize ***slow*** and ***fast*** at the top of the list.
- As long as the two pointers point to different positions, we leave the ***slow*** where it is and advance the ***fast***.
- When they meet, we move the ***fast*** at the top of the list and we move the ***slow*** one position down the list.

TFM With Identical Clause Elimination

1. $\{p,q\}$	Premise
2. $\{\neg p,r\}$	Premise
3. $\{\neg q, r\}$	Premise
4. $\{\neg r\}$	Premise
5. $\{q,r\}$	1,2
6. $\{p,r\}$	1,3
7. $\{\neg p\}$	2,4
8. $\{\neg q\}$	3,4
9. $\{r\}$	3,5
10. $\{q\}$	4,5
11. $\{p\}$	4,6
12. $\{\}$	4,9

Another TFM example

1. $\{p,q\}$	
2. $\{p,\neg q\}$	
3. $\{\neg p,q\}$	
4. $\{\neg p,\neg q\}$	
5. $\{p\}$	1,2
6. $\{q\}$	1,3
7. $\{\neg q, q\}$	2,3
8. $\{p,\neg p\}$	2,3
9. $\{q,\neg q\}$	1,4
9.5 $\{p,\neg p\}$	1,4
10. $\{\neg q\}$	2,4
11. $\{\neg p\}$	3, 4
12. $\{q\}$	3,5
13. $\{\neg q\}$	4,5
14. $\{p\}$	2,6
15. $\{\neg p\}$	4,6
16. $\{p,q\}$	1,7
17. $\{\neg q, p\}$	2,7
18. $\{\neg p,q\}$	3,7
19. $\{\neg q,\neg p\}$	4,7
20. $\{q\}$	6,7

continued

21. $\{\neg q, q\}$ 7, 7
22. $\{\neg q, q\}$ 7, 7
23. $\{q, p\}$ 1,8
24. $\{\neg q, p\}$ 2,8
25. $\{\neg p, q\}$ 3,8
26. $\{\neg p, \neg q\}$ 4,8
27. $\{p\}$ 5,8
28. $\{\neg p, p\}$ 8,8
29. $\{\neg p, p\}$ 8,8
30. $\{p, q\}$ 1,9
31. $\{\neg q, p\}$ 2,9
32. $\{\neg p, q\}$ 3,9
33. $\{\neg q, \neg p\}$ 4, 9
34. $\{q\}$ 6,9
35. $\{\neg q, q\}$ 7,9
36. $\{q, \neg q\}$ 9,9
37. $\{q, \neg q\}$ 9,9
38. $\{p\}$ 1,10
39. $\{\neg p\}$ 3,10
40. $\{\}$ 6,10

Proof With Identical Clause Elimination

1. $\{p,q\}$
2. $\{p,\neg q\}$
3. $\{\neg p,q\}$
4. $\{\neg p,\neg q\}$
5. $\{p\}$ 1,2
6. $\{q\}$ 1,3
7. $\{\neg q, q\}$ 2,3
8. $\{p,\neg p\}$ 2,3
9. $\{q,\neg q\}$ 1,4
10. $\{\neg q\}$ 2,4
11. $\{\neg p\}$ 3,4
12. $\{\}$ 6,10

Tautology Elimination

A *tautology* is a clause with a complementary pair of literals.

1. $\{p,q\}$ Premise
2. $\{p,\neg p\}$ Premise
3. $\{p,q\}$ 1,2 (not a new thing, we already knew it)

We can remove tautologies, such as $\{p,\neg p\}$, without causing any harm.

Proof with TE and ICE

1. $\{p, q\}$
2. $\{p, \neg q\}$
3. $\{\neg p, q\}$
4. $\{\neg p, \neg q\}$
5. $\{p\}$ 1,2
6. $\{q\}$ 1,3
7. $\{\neg q\}$ 2, 4
8. $\{\neg p\}$ 3,4
9. $\{\}$ 6,7