Propositional and First-Order Logic

Chapter 7.4─7.8, 8.1─8.3, 8.5

Some material adopted from notes
by Andreas Geyer-Schulz
and Chuck Dyer

Overview

- Propositional logic (quick review)
- Problems with propositional logic
- First-order logic (review)
  - Properties, relations, functions, quantifiers, …
  - Terms, sentences, wffs, axioms, theories, proofs, …
- Extensions to first-order logic
- Logical agents
  - Reflex agents
  - Representing change: situation calculus, frame problem
  - Preferences on actions
  - Goal-based agents

Propositional Logic: Review

Propositional logic

- **Logical constants**: true, false
- **Propositional symbols**: P, Q, S, ... (**atomic sentences**)
- Wrapping **parentheses**: (…)
- Sentences are combined by **connectives**:

$\wedge$ ...and	[conjunction]
$\vee$ ...or	[disjunction]
$\Rightarrow$...implies	[implication / conditional]
$\Leftrightarrow$..is equivalent	[biconditional]
$\neg$ ...not	[negation]

- **Literal**: atomic sentence or negated atomic sentence
  - P, $\neg$ P

Examples of PL sentences

- $(P \wedge Q) \rightarrow R$
  "If it is hot and humid, then it is raining"
- $Q \rightarrow P$
  "If it is humid, then it is hot"
- Q
  "It is humid."
- A better way:
  Ho = "It is hot"
  Hu = "It is humid"
  R = "It is raining"

Propositional logic (PL)

- A simple language useful for showing key ideas and definitions
- User defines a set of propositional symbols, like P and Q.
- User defines the **semantics** of each propositional symbol:
  - P means "It is hot"
  - Q means "It is humid"
  - R means "It is raining"
- A sentence (well formed formula) is defined as follows:
  - A symbol is a sentence
  - If S is a sentence, then $\neg S$ is a sentence
  - If S is a sentence, then (S) is a sentence
  - If S and T are sentences, then $(S \vee T)$, $(S \wedge T)$, $(S \rightarrow T)$, and $(S \leftrightarrow T)$ are sentences
  - A sentence results from a finite number of applications of the above rules

A BNF grammar of sentences in propositional logic

```
S := <Sentence> ;
<Sentence> := <AtomicSentence> | <ComplexSentence> ;
<AtomicSentence> := "TRUE" | "FALSE" |
                    "P" | "Q" | "S" ;
<ComplexSentence> := "(" <Sentence> ")" |
          <Sentence> <Connective> <Sentence> |
          "NOT" <Sentence> ;
<Connective> := "AND" | "OR" | "IMPLIES" |
  "EQUIVALENT" ;
```

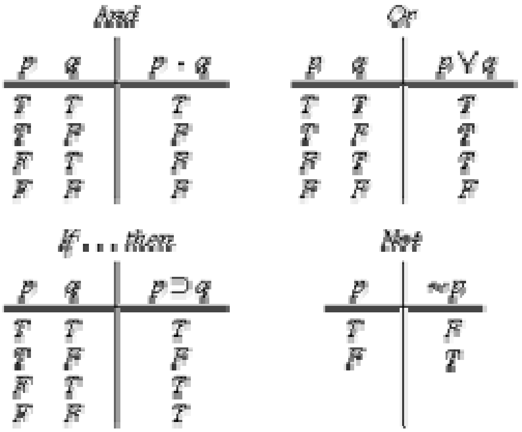
Some terms

- The meaning or **semantics** of a sentence determines its **interpretation**.
- Given the truth values of all symbols in a sentence, it can be "evaluated" to determine its **truth value** (True or False).
- A **model** for a KB is a "possible world" (assignment of truth values to propositional symbols) in which each sentence in the KB is True.

More terms

- A **valid sentence** or **tautology** is a sentence that is True under all interpretations, no matter what the world is actually like or what the semantics is. Example: “It’s raining or it’s not raining.”
- An **inconsistent sentence** or **contradiction** is a sentence that is False under all interpretations. The world is never like what it describes, as in “It’s raining and it’s not raining.”
- **P entails Q**, written P |= Q, means that whenever P is True, so is Q. In other words, all models of P are also models of Q.

Truth tables



And

p	q	$p \cdot q$
T	T	T
T	F	F
F	T	F
F	F	F

Or

p	q	$p \vee q$
T	T	T
T	F	T
F	T	T
F	F	F

If . . . then

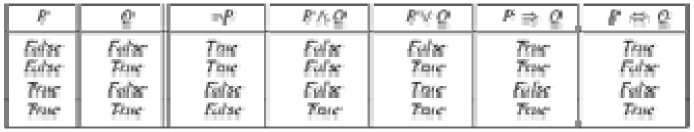
p	q	$p \supset q$
T	T	T
T	F	F
F	T	T
F	F	T

Not

p	$\sim p$
T	F
F	T

Truth tables II

The five logical connectives:

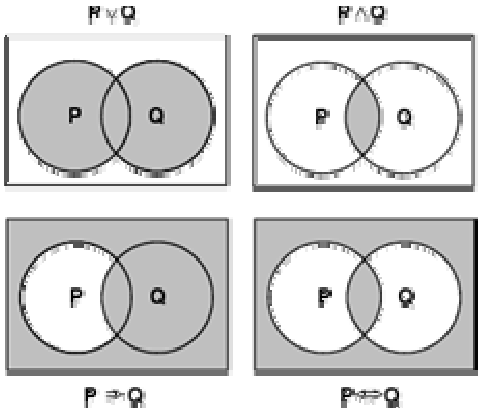


P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
False	False	True	False	False	True	True
False	True	True	False	True	True	False
True	False	False	False	True	False	False
True	True	False	True	True	True	True

A complex sentence:

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

Models of complex sentences



Inference rules

- **Logical inference** is used to create new sentences that logically follow from a given set of predicate calculus sentences (KB).
- An inference rule is **sound** if every sentence X produced by an inference rule operating on a KB logically follows from the KB. (That is, the inference rule does not create any contradictions)
- An inference rule is **complete** if it is able to produce every expression that logically follows from (is entailed by) the KB. (Note the analogy to complete search algorithms.)

Sound rules of inference

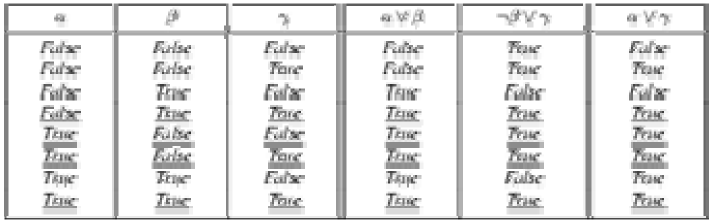
- Here are some examples of sound rules of inference
  - *A rule is sound if its conclusion is true whenever the premise is true*
- Each can be shown to be sound using a truth table

RULE	PREMISE	CONCLUSION
Modus Ponens	A, A → B	B
And Introduction	A, B	A ∧ B
And Elimination	A ∧ B	A
Double Negation	¬¬A	A
Unit Resolution	A ∨ B, ¬B	A
Resolution	**A ∨ B, ¬B ∨ C**	**A ∨ C**

Soundness of modus ponens

A	B	A → B	OK?
True	True	True	√
True	False	False	√
False	True	True	√
False	False	True	√

Soundness of the resolution inference rule



α	β	γ	$\alpha \vee \beta$	$\neg\beta \vee \gamma$	$\alpha \vee \gamma$
False	False	False	False	True	False
False	False	True	False	True	True
False	True	False	True	False	False
False	True	True	True	True	True
True	False	False	True	True	True
True	False	True	True	True	True
True	True	False	True	False	True
True	True	True	True	True	True

Proving things

- A **proof** is a sequence of sentences, where each sentence is either a premise or a sentence derived from earlier sentences in the proof by one of the rules of inference.
- The last sentence is the **theorem** (also called goal or query) that we want to prove.
- Example for the "weather problem" given above.

1 Hu	Premise	"It is humid"
2 $Hu \rightarrow Ho$	Premise	"If it is humid, it is hot"
3 Ho	Modus Ponens(1,2)	"It is hot"
4 $(Ho \wedge Hu) \rightarrow R$	Premise	"If it's hot & humid, it's raining"
5 $Ho \wedge Hu$	And Introduction(1,3)	"It is hot and humid"
6 R	Modus Ponens(4,5)	"It is raining"

Horn sentences

- A **Horn sentence** or **Horn clause** has the form:

$$P1 \wedge P2 \wedge P3 \ldots \wedge Pn \rightarrow Q$$

or alternatively

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

$$\neg P1 \vee \neg P2 \vee \neg P3 \ldots \vee \neg Pn \vee Q$$

where Ps and Q are non-negated atoms

- To get a proof for Horn sentences, apply Modus Ponens repeatedly until nothing can be done
- We will use the Horn clause form later

Entailment and derivation

- **Entailment: KB |= Q**
  - Q is entailed by KB (a set of premises or assumptions) if and only if there is no logically possible world in which Q is false while all the premises in KB are true.
  - Or, stated positively, Q is entailed by KB if and only if the conclusion is true in every logically possible world in which all the premises in KB are true.
- **Derivation: KB |- Q**
  - We can derive Q from KB if there is a proof consisting of a sequence of valid inference steps starting from the premises in KB and resulting in Q

Two important properties for inference

Soundness: If KB |- Q then KB |= Q

- If Q is derived from a set of sentences KB using a given set of rules of inference, then Q is entailed by KB.
- Hence, inference produces only real entailments, or any sentence that follows deductively from the premises is valid.

Completeness: If KB |= Q then KB |- Q

- If Q is entailed by a set of sentences KB, then Q can be derived from KB using the rules of inference.
- Hence, inference produces all entailments, or all valid sentences can be proved from the premises.

Problems with Propositional Logic

Propositional logic is a weak language

- Hard to identify “individuals” (e.g., Mary, 3)
- Can’t directly talk about properties of individuals or relations between individuals (e.g., “Bill is tall”)
- Generalizations, patterns, regularities can’t easily be represented (e.g., “all triangles have 3 sides”)
- First-Order Logic (abbreviated FOL or FOPC) is expressive enough to concisely represent this kind of information

  FOL adds relations, variables, and quantifiers, e.g.,
  - *“Every elephant is gray”:* ∀ x (elephant(x) → gray(x))
  - *“There is a white alligator”:* ∃ x (alligator(X) ^ white(X))

Example

- Consider the problem of representing the following information:
  - Every person is mortal.
  - Confucius is a person.
  - Confucius is mortal.
- How can these sentences be represented so that we can infer the third sentence from the first two?

Example II

- In PL we have to create propositional symbols to stand for all or part of each sentence. For example, we might have:

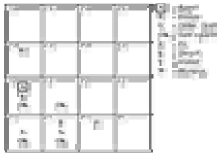
  P = “person”; Q = “mortal”; R = “Confucius”
- so the above 3 sentences are represented as:

  $P \rightarrow Q$; $R \rightarrow P$; $R \rightarrow Q$
- Although the third sentence is entailed by the first two, we needed an explicit symbol, R, to represent an individual, Confucius, who is a member of the classes “person” and “mortal”
- To represent other individuals we must introduce separate symbols for each one, with some way to represent the fact that all individuals who are “people” are also “mortal”

The “Hunt the Wumpus” agent

- Some atomic propositions:

  S12 = There is a stench in cell (1,2)
  B34 = There is a breeze in cell (3,4)
  W22 = The Wumpus is in cell (2,2)
  V11 = We have visited cell (1,1)
  OK11 = Cell (1,1) is safe.
  etc



- Some rules:

  (R1) ¬S11 → ¬W11 ∧ ¬ W12 ∧ ¬ W21
  (R2) ¬ S21 → ¬W11 ∧ ¬ W21 ∧ ¬ W22 ∧ ¬ W31
  (R3) ¬ S12 → ¬W11 ∧ ¬ W12 ∧ ¬ W22 ∧ ¬ W13
  (R4) S12 → W13 ∨ W12 ∨ W22 ∨ W11
  Etc.

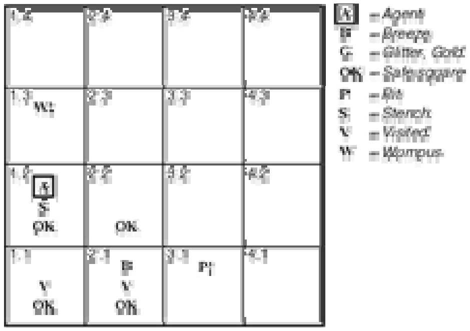
- Note that the lack of variables requires us to give similar rules for each cell

---

After the third move

We can prove that the Wumpus is in (1,3) using the four rules given.

See R&N section 7.5



Proving W13

- Apply MP with ¬S11 and R1:

  ¬ W11 ∧ ¬ W12 ∧ ¬ W21
- Apply And-Elimination to this, yielding 3 sentences:

  ¬ W11, ¬ W12, ¬ W21
- Apply MP to ~S21 and R2, then apply And-elimination:

  ¬ W22, ¬ W21, ¬ W31
- Apply MP to S12 and R4 to obtain:

  W13 ∨ W12 ∨ W22 ∨ W11
- Apply Unit resolution on (W13 ∨ W12 ∨ W22 ∨ W11) and ¬W11:

  W13 ∨ W12 ∨ W22
- Apply Unit Resolution with (W13 ∨ W12 ∨ W22) and ¬W22:

  W13 ∨ W12
- Apply UR with (W13 ∨ W12) and ¬W12:

  W13
- QED

Problems with the propositional Wumpus hunter

- Lack of variables prevents stating more general rules
  - We need a set of similar rules for each cell
- Change of the KB over time is difficult to represent
  - Standard technique is to index facts with the time when they’re true
  - This means we have a separate KB for every time point

Propositional logic: Summary

- The process of deriving new sentences from old one is called **inference**.
  - **Sound** inference processes derives true conclusions given true premises
  - **Complete** inference processes derive all true conclusions from a set of premises
- A **valid sentence** is true in all worlds under all interpretations
- If an implication sentence can be shown to be valid, then—given its premise—its consequent can be derived
- Different logics make different **commitments** about what the world is made of and what kind of beliefs we can have regarding the facts
  - Logics are useful for the commitments they do not make because lack of commitment gives the knowledge base engineer more freedom
- **Propositional logic** commits only to the existence of facts that may or may not be the case in the world being represented
  - It has a simple syntax and simple semantics. It suffices to illustrate the process of inference
  - Propositional logic quickly becomes impractical, even for very small worlds

First-Order Logic: Review

First-order logic

- First-order logic (FOL) models the world in terms of
  - **Objects,** which are things with individual identities
  - **Properties** of objects that distinguish them from other objects
  - **Relations** that hold among sets of objects
  - **Functions,** which are a subset of relations where there is only one “value” for any given “input”
- Examples:
  - Objects: Students, lectures, companies, cars ...
  - Relations: Brother-of, bigger-than, outside, part-of, has-color, occurs-after, owns, visits, precedes, ...
  - Properties: blue, oval, even, large, ...
  - Functions: father-of, best-friend, second-half, one-more-than ...

User provides

- **Constant symbols,** which represent individuals in the world
  - Mary
  - 3
  - Green
- **Function symbols,** which map individuals to individuals
  - father-of(Mary) = John
  - color-of(Sky) = Blue
- **Predicate symbols,** which map individuals to truth values
  - greater(5,3)
  - green(Grass)
  - color(Grass, Green)

FOL Provides

- **Variable symbols**
  - E.g., x, y, foo
- **Connectives**
  - Same as in PL: not (¬), and (∧), or (∨), implies (→), if and only if (biconditional ↔)
- **Quantifiers**
  - Universal ∀**x** or **(Ax)**
  - Existential ∃**x** or **(Ex)**

Sentences are built from terms and atoms

- A **term** (denoting a real-world individual) is a constant symbol, a variable symbol, or an n-place function of n terms.
  - x and f(x_1, ..., x_n) are terms, where each x_i is a term.
  - A term with no variables is a **ground term**
- An **atomic sentence** (which has value true or false) is an n-place predicate of n terms
- A **complex sentence** is formed from atomic sentences connected by the logical connectives:
  - ¬P, P∨Q, P∧Q, P→Q, P↔Q where P and Q are sentences
- A **quantified sentence** adds quantifiers ∀ and ∃
- A **well-formed formula (wff)** is a sentence containing no “free” variables. That is, all variables are “bound” by universal or existential quantifiers.
  - (∀x)P(x,y) has x bound as a universally quantified variable, but y is free.

A BNF for FOL

```
S := <Sentence> ;
<Sentence> := <AtomicSentence> |
          <Sentence> <Connective> <Sentence> |
          <Quantifier> <Variable>,... <Sentence> |
          "NOT" <Sentence> |
          "(" <Sentence> ")";
<AtomicSentence> := <Predicate> "(" <Term>, ... ")" |
                    <Term> "=" <Term>;
<Term> := <Function> "(" <Term>, ... ")" |
          <Constant> |
          <Variable>;
<Connective> := "AND" | "OR" | "IMPLIES" | "EQUIVALENT";
<Quantifier> := "EXISTS" | "FORALL" ;
<Constant> := "A" | "X1" | "John" | ... ;
<Variable> := "a" | "x" | "s" | ... ;
<Predicate> := "Before" | "HasColor" | "Raining" | ... ;
<Function> := "Mother" | "LeftLegOf" | ... ;
```

Quantifiers

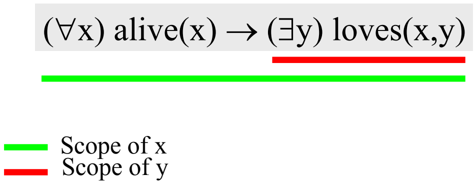
- **Universal quantification**
  - (∀x)P(x) means that P holds for **all** values of x in the domain associated with that variable
  - E.g., (∀x) dolphin(x) → mammal(x)
- **Existential quantification**
  - (∃ x)P(x) means that P holds for **some** value of x in the domain associated with that variable
  - E.g., (∃ x) mammal(x) ∧ lays-eggs(x)
  - Permits one to make a statement about some object without naming it

Quantifiers

- Universal quantifiers are often used with “implies” to form “rules”:
  (∀x) student(x) → smart(x) means “All students are smart”
- Universal quantification is *rarely* used to make blanket statements about every individual in the world:
  (∀x)student(x)∧smart(x) means “Everyone in the world is a student and is smart”
- Existential quantifiers are usually used with “and” to specify a list of properties about an individual:
  (∃x) student(x) ∧ smart(x) means “There is a student who is smart”
- A common mistake is to represent this English sentence as the FOL sentence:
  (∃x) student(x) → smart(x)
  - But what happens when there is a person who is *not* a student?

Quantifier Scope

- FOL sentences have structure, like programs
- In particular, the variables in a sentence have a scope
- For example, suppose we want to say
  - “everyone who is alive loves someone”
  - (∀x) alive(x) → (∃y) loves(x,y)
- Here’s how we soce the variables



(∀x) alive(x) → (∃y) loves(x,y)

🟩 Scope of x
🟥 Scope of y

Quantifier Scope

- **Switching the order of universal quantifiers *does not* change the meaning:**
  - (∀x)(∀y)P(x,y) ↔ (∀y)(∀x) P(x,y)
  - “Dogs hate cats”.
- **Similarly, you can switch the order of existential quantifiers:**
  - (∃x)(∃y)P(x,y) ↔ (∃y)(∃x) P(x,y)
  - “A cat killed a dog”
- **Switching the order of universals and existentials *does* change meaning:**
  - Everyone likes someone: (∀x)(∃y) likes(x,y)
  - Someone is liked by everyone: (∃y)(∀x) likes(x,y)

Connections between All and Exists

- **We can relate sentences involving ∀ and ∃ using De Morgan’s laws:**
  1. (∀x) ¬P(x) ↔ ¬(∃x) P(x)
  2. ¬(∀x) P ↔ (∃x) ¬P(x)
  3. (∀x) P(x) ↔ ¬ (∃x) ¬P(x)
  4. (∃x) P(x) ↔ ¬(∀x) ¬P(x)
- **Examples**
  1. All dogs don’t like cats ↔ No dog’s like cats
  2. Not all dogs dance ↔ There is a dog that doesn’t dance.
  3. All dogs sleep ↔ There is no dog that doesn’t sleep
  4. There is a dog that talks ↔ Not all dogs can’t talk

Quantified inference rules

- Universal instantiation
  - $\forall x\ P(x)\ \therefore\ P(A)$
- Universal generalization
  - $P(A) \wedge P(B) \ldots \therefore\ \forall x\ P(x)$
- Existential instantiation
  - $\exists x\ P(x)\ \therefore P(F)$ ← **skolem constant F**
- Existential generalization
  - $P(A)\ \therefore\ \exists x\ P(x)$

Universal instantiation
(a.k.a. universal elimination)

- If $(\forall x)\ P(x)$ is true, then P(C) is true, where C is *any* constant in the domain of x
- Example:

  $(\forall x)$ eats(Ziggy, x) ⇒ eats(Ziggy, IceCream)
- The variable symbol can be replaced by any ground term, i.e., any constant symbol or function symbol applied to ground terms only

Existential instantiation
(a.k.a. existential elimination)

- From $(\exists x)\ P(x)$ infer P(c)
- Example:
  - $(\exists x)$ eats(Ziggy, x) → eats(Ziggy, Stuff)
- Note that the variable is replaced by a **brand-new constant** not occurring in this or any other sentence in the KB
- Also known as skolemization; constant is a **skolem constant**
- In other words, we don’t want to accidentally draw other inferences about it by introducing the constant
- Convenient to use this to reason about the unknown object, rather than constantly manipulating the existential quantifier

Existential generalization
(a.k.a. existential introduction)

- If P(c) is true, then $(\exists x)\ P(x)$ is inferred.
- Example

  eats(Ziggy, IceCream) ⇒ $(\exists x)$ eats(Ziggy, x)
- All instances of the given constant symbol are replaced by the new variable symbol
- Note that the variable symbol cannot already exist anywhere in the expression

Translating English to FOL

- **Every gardener likes the sun.**

  $\forall x\ \text{gardener}(x) \rightarrow \text{likes}(x,\text{Sun})$
- **You can fool some of the people all of the time.**

  $\exists x\ \forall t\ \ \text{person}(x) \wedge \text{time}(t) \rightarrow \text{can-fool}(x,t)$
- **You can fool all of the people some of the time.** (two ways)

  $\forall x\ \exists t\ (\text{person}(x) \rightarrow \text{time}(t) \wedge \text{can-fool}(x,t))$

  $\forall x\ (\text{person}(x) \rightarrow \exists t\ (\text{time}(t) \wedge \text{can-fool}(x,t))$
- **All purple mushrooms are poisonous**.

  $\forall x\ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \text{poisonous}(x)$

Translating English to FOL

- **No purple mushroom is poisonous.** (two ways)

  $\neg\exists x\ \text{purple}(x) \wedge \text{mushroom}(x) \wedge \text{poisonous}(x)$

  $\forall x\ \ (\text{mushroom}(x) \wedge \text{purple}(x)) \rightarrow \neg\text{poisonous}(x)$
- **There are exactly two purple mushrooms**.

  $\exists x\ \exists y\ \text{mushroom}(x) \wedge \text{purple}(x) \wedge \text{mushroom}(y) \wedge \text{purple}(y)\ {}^\wedge$
  $\neg(x=y) \wedge \forall z\ (\text{mushroom}(z) \wedge \text{purple}(z)) \rightarrow ((x=z) \vee (y=z))$
- **Bush is not tall.**

  $\neg\text{tall}(\text{Bush})$
- **X is above Y iff X is on directly on top of Y or there is a pile of one or more other objects directly on top of one another starting with X and ending with Y.**

  $\forall x\ \forall y\ \text{above}(x,y) \leftrightarrow (\text{on}(x,y) \vee\ \ \exists z\ (\text{on}(x,z) \wedge \text{above}(z,y)))$

Logic and People

“Logic—the last refuge of a scoundrel.”

- People can easily be confused by logic
- And are often suspicious of it, or give it too much weight

Monty Python example (Russell & Norvig)

FIRST VILLAGER: We have found a witch. May we burn her?
ALL: A witch! Burn her!
BEDEVERE: Why do you think she is a witch?
SECOND VILLAGER: She turned *me* into a newt.
B: A newt?
V2 *(after looking at himself for some time)*: I got better.
ALL: Burn her anyway.
B: Quiet! Quiet! There are ways of telling whether she is a witch.

B: Tell me… what do you do with witches?

ALL: Burn them!

B: And what do you burn, apart from witches?

V4: …wood?

B: So **why do witches burn**?

V2 *(pianissimo)*: **because they're made of wood?**

B: Good.

ALL: I see. Yes, of course.

B: So how can we tell if she is made of wood?

V1: Make a bridge out of her.

B: Ah… but can you not also make bridges out of stone?

ALL: Yes, of course… um… er…

B: Does wood sink in water?

ALL: No, no, it floats. Throw her in the pond.

B: Wait. Wait… tell me, what also floats on water?

ALL: Bread? No, no no. Apples… gravy… very small rocks…

B: No, no, no,

KING ARTHUR: A duck!

(They all turn and look at Arthur. Bedevere looks up, very impressed.)

B: Exactly. So… logically…

V1 *(beginning to pick up the thread)*: **If she… weighs the same as a duck… she's made of wood**.

B: And therefore?

ALL: **A witch!**

Monty Python Fallacy #1

- $\forall x\ \text{witch}(x) \rightarrow \text{burns}(x)$
- $\forall x\ \text{wood}(x) \rightarrow \text{burns}(x)$
- -------------------------------
- $\therefore\ \forall z\ \text{witch}(x) \rightarrow \text{wood}(x)$

- $p \rightarrow q$
- $r \rightarrow q$
- ---------
- $p \rightarrow r$

Fallacy: Affirming the conclusion

Monty Python Near-Fallacy #2

wood(x) → can-build-bridge(x)

∴ can-build-bridge(x) → wood(x)

- B: Ah… but can you not also make bridges out of stone?

Monty Python Fallacy #3

- ∀x wood(x) → floats(x)
- ∀x duck-weight (x) → floats(x)
- -------------------------------
- ∴ ∀x duck-weight(x) → wood(x)

- p → q
- r → q
- -----------
- ∴ r → p

Monty Python Fallacy #4

- ∀z light(z) → wood(z)
- light(W)
- ------------------------------
- ∴ wood(W) ok…………..

- witch(W) → wood(W) applying universal instan. to fallacious conclusion #1
- wood(W)
- ---------------------------------
- ∴ witch(z)

Example: A simple genealogy KB by FOL

- **Build a small genealogy knowledge base using FOL that**
  - contains facts of immediate family relations (spouses, parents, etc.)
  - contains definitions of more complex relations (ancestors, relatives)
  - is able to answer queries about relationships between people
- **Predicates:**
  - parent(x, y), child(x, y), father(x, y), daughter(x, y), etc.
  - spouse(x, y), husband(x, y), wife(x,y)
  - ancestor(x, y), descendant(x, y)
  - male(x), female(y)
  - relative(x, y)
- **Facts:**
  - husband(Joe, Mary), son(Fred, Joe)
  - spouse(John, Nancy), male(John), son(Mark, Nancy)
  - father(Jack, Nancy), daughter(Linda, Jack)
  - daughter(Liz, Linda)
  - etc.

- **Rules for genealogical relations**
  - (∀x,y) parent(x, y) ↔ child (y, x)
    (∀x,y) father(x, y) ↔ parent(x, y) ∧ male(x) (similarly for mother(x, y))
    (∀x,y) daughter(x, y) ↔ child(x, y) ∧ female(x) (similarly for son(x, y))
  - (∀x,y) husband(x, y) ↔ spouse(x, y) ∧ male(x) (similarly for wife(x, y))
    (∀x,y) spouse(x, y) ↔ spouse(y, x) (**spouse relation is symmetric**)
  - (∀x,y) parent(x, y) → ancestor(x, y)
    (∀x,y)(∃z) parent(x, z) ∧ ancestor(z, y) → ancestor(x, y)
  - (∀x,y) descendant(x, y) ↔ ancestor(y, x)
  - (∀x,y)(∃z) ancestor(z, x) ∧ ancestor(z, y) → relative(x, y)
    (related by common ancestry)
    (∀x,y) spouse(x, y) → relative(x, y) (related by marriage)
    (∀x,y)(∃z) relative(z, x) ∧ relative(z, y) → relative(x, y) (**transitive**)
    (∀x,y) relative(x, y) ↔ relative(y, x) (**symmetric**)
- **Queries**
  - ancestor(Jack, Fred) /* the answer is yes */
  - relative(Liz, Joe) /* the answer is yes */
  - relative(Nancy, Matthew)
    /* no answer in general, no if under closed world assumption */
  - (∃z) ancestor(z, Fred) ∧ ancestor(z, Liz)

Axioms for Set Theory in FOL

1. The only sets are the empty set and those made by adjoining something to a set:
   ∀s set(s) <=> (s=EmptySet) v (∃x,r Set(r) ^ s=Adjoin(s,r))
2. The empty set has no elements adjoined to it:
   ~ ∃x,s Adjoin(x,s)=EmptySet
3. Adjoining an element already in the set has no effect:
   ∀x,s Member(x,s) <=> s=Adjoin(x,s)
4. The only members of a set are the elements that were adjoined into it:
   ∀x,s Member(x,s) <=> ∃y,r (s=Adjoin(y,r) ^ (x=y ∨ Member(x,r)))
5. A set is a subset of another iff all of the 1st set's members are members of the 2nd:
   ∀s,r Subset(s,r) <=> (∀x Member(x,s) => Member(x,r))
6. Two sets are equal iff each is a subset of the other:
   ∀s,r (s=r) <=> (subset(s,r) ^ subset(r,s))
7. Intersection
   ∀x,s1,s2 member(X,intersection(S1,S2)) <=> member(X,s1) ^ member(X,s2)
8. Union
   ∃x,s1,s2 member(X,union(s1,s2)) <=> member(X,s1) ∨ member(X,s2)

Semantics of FOL

- **Domain M:** the set of all objects in the world (of interest)
- **Interpretation I:** includes
  - Assign each constant to an object in M
  - Define each function of n arguments as a mapping M^n => M
  - Define each predicate of n arguments as a mapping M^n => {T, F}
  - Therefore, every ground predicate with any instantiation will have a truth value
  - In general there is an infinite number of interpretations because |M| is infinite
- **Define logical connectives:** **~, ^, v, =>, <=>** as in PL
- **Define semantics of (∀x) and (∃x)**
  - (∀x) P(x) is true iff P(x) is true under all interpretations
  - (∃x) P(x) is true iff P(x) is true under some interpretation

---

- **Model:** an interpretation of a set of sentences such that every sentence is *True*
- **A sentence is**
  - **satisfiable** if it is true under some interpretation
  - **valid** if it is true under all possible interpretations
  - **inconsistent** if there does not exist any interpretation under which the sentence is true
- **Logical consequence:** S |= X if all models of S are also
  models of X

Axioms, definitions and theorems

- **Axioms** are facts and rules that attempt to capture all of the (important) facts and concepts about a domain; axioms can be used to prove **theorems**
  - –Mathematicians don’t want any unnecessary (dependent) axioms –ones that can be derived from other axioms
  - –Dependent axioms can make reasoning faster, however
  - –Choosing a good set of axioms for a domain is a kind of design problem
- A **definition** of a predicate is of the form “p(X) ↔ …” and can be decomposed into two parts
  - –**Necessary** description: “p(x) → …”
  - –**Sufficient** description “p(x) ← …”
  - –Some concepts don’t have complete definitions (e.g., person(x))

More on definitions

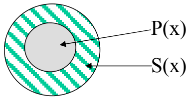
- Examples: define father(x, y) by parent(x, y) and male(x)
  - parent(x, y) is a necessary (**but not sufficient**) description of
    father(x, y)
    - father(x, y) → parent(x, y)
  - parent(x, y) ^ male(x) ^ age(x, 35) is a **sufficient** (**but not necessary**) description of father(x, y):

    father(x, y) ← parent(x, y) ^ male(x) ^ age(x, 35)
  - parent(x, y) ^ male(x) is a **necessary and sufficient** description of father(x, y)

    parent(x, y) ^ male(x) ↔ father(x, y)

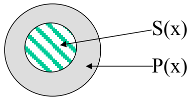
More on definitions

S(x) is a
necessary
condition of P(x)



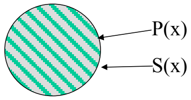
(∀x) P(x) => S(x)

S(x) is a
sufficient
condition of P(x)



(∀x) P(x) <= S(x)

S(x) is a
necessary and
sufficient
condition of P(x)



(∀x) P(x) <=> S(x)

Higher-order logic

- FOL only allows to quantify over variables, and variables can only range over objects.
- HOL allows us to quantify over relations
- Example: (quantify over functions)

  “two functions are equal iff they produce the same value for all arguments”

  ∀f ∀g (f = g) ↔ (∀x f(x) = g(x))
- Example: (quantify over predicates)

  ∀r transitive(r) → (∀xyz) r(x,y) ∧ r(y,z) → r(x,z))
- More expressive, but undecidable.

Expressing uniqueness

- Sometimes we want to say that there is a single, unique object that satisfies a certain condition
- "There exists a unique x such that king(x) is true"
  - $\exists x\ king(x) \wedge \forall y\ (king(y) \rightarrow x=y)$
  - $\exists x\ king(x) \wedge \neg\exists y\ (king(y) \wedge x \neq y)$
  - $\exists!\ x\ king(x)$
- "Every country has exactly one ruler"
  - $\forall c\ country(c) \rightarrow \exists!\ r\ ruler(c,r)$
- Iota operator: "ι x P(x)" means "the unique x such that p(x) is true"
  - "The unique ruler of Freedonia is dead"
  - $dead(\iota\ x\ ruler(freedonia,x))$

Notational differences

- **Different symbols** for *and, or, not, implies, ...*
  - $\forall\ \exists\ \Rightarrow\ \Leftrightarrow\ \wedge\ \vee\ \neg\ \bullet\ \supset$
  - p v (q ^ r)
  - p + (q * r)
  - etc
- **Prolog**

```
cat(X) :- furry(X), meows (X), has(X, claws)
```

- **Lispy notations**

```
(forall ?x (implies (and (furry ?x)
                          (meows ?x)
                          (has ?x claws))
                     (cat ?x)))
```

Logical Agents

Logical agents for the Wumpus World

Three (non-exclusive) agent architectures:

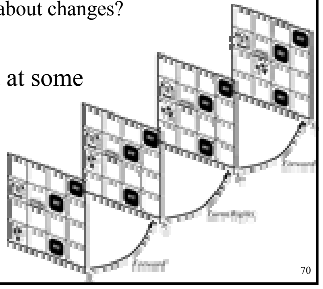
- Reflex agents
  - Have rules that classify situations, specifying how to react to each possible situation
- Model-based agents
  - Construct an internal model of their world
- Goal-based agents
  - Form goals and try to achieve them

A simple reflex agent

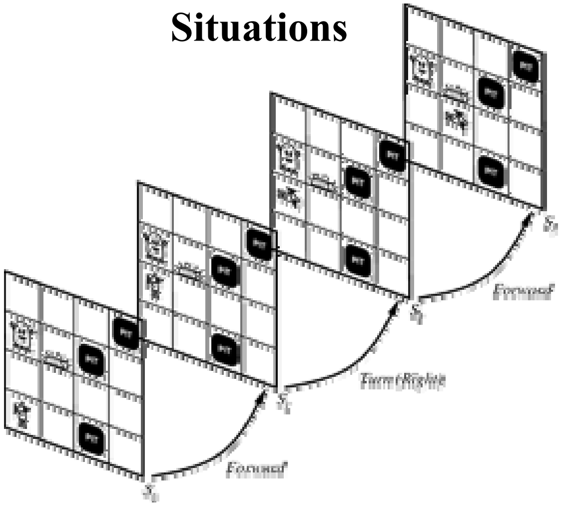
- Rules to map percepts into observations:
  - ∀b,g,u,c,t Percept([Stench, b, g, u, c], t) → Stench(t)
  - ∀s,g,u,c,t Percept([s, Breeze, g, u, c], t) → Breeze(t)
  - ∀s,b,u,c,t Percept([s, b, Glitter, u, c], t) → AtGold(t)
- Rules to select an action given observations:
  - ∀t AtGold(t) → Action(Grab, t);
- Some difficulties:
  - Consider Climb. There is no percept that indicates the agent should climb out – position and holding gold are not part of the percept sequence
  - Loops – the percept will be repeated when you return to a square, which should cause the same response (unless we maintain some internal model of the world)

Representing change

- Representing change in the world in logic can be tricky.
- One way is just to change the KB
  - Add and delete sentences from the KB to reflect changes
  - How do we remember the past, or reason about changes?
- **Situation calculus** is another way
- A **situation** is a snapshot of the world at some instant in time
- When the agent performs an action A in situation S1, the result is a new situation S2.



Situations



Situation calculus

- A **situation** is a snapshot of the world at an interval of time during which nothing changes
- Every true or false statement is made with respect to a particular situation.
  - Add **situation variables** to every predicate.
  - at(Agent,1,1) becomes at(Agent,1,1,s0): at(Agent,1,1) is true in situation (i.e., state) s0.
  - Alernatively, add a special 2nd-order predicate, **holds(f,s),** that means “f is true in situation s.” E.g., holds(at(Agent,1,1),s0)
- Add a new function, **result(a,s),** that maps a situation s into a new situation as a result of performing action a. For example, result(forward, s) is a function that returns the successor state (situation) to s
- Example: The action agent-walks-to-location-y could be represented by
  - (∀x)(∀y)(∀s) (at(Agent,x,s) ∧ ¬onbox(s)) → at(Agent,y,result(walk(y),s))

Deducing hidden properties

- From the perceptual information we obtain in situations, we can **infer properties of locations**

  $\forall l,s\ at(Agent,l,s) \wedge Breeze(s) \rightarrow Breezy(l)$

  $\forall l,s\ at(Agent,l,s) \wedge Stench(s) \rightarrow Smelly(l)$
- Neither Breezy nor Smelly need situation arguments because pits and Wumpuses do not move around

Deducing hidden properties II

- We need to write some rules that relate various aspects of a single world state (as opposed to across states)
- There are two main kinds of such rules:
  - **Causal rules** reflect the assumed direction of causality in the world:

    $(\forall l1,l2,s)\ At(Wumpus,l1,s) \wedge Adjacent(l1,l2) \rightarrow Smelly(l2)$

    $(\forall\ l1,l2,s)\ At(Pit,l1,s) \wedge Adjacent(l1,l2) \rightarrow Breezy(l2)$

    Systems that reason with causal rules are called **model-based reasoning systems**
  - **Diagnostic rules** infer the presence of **hidden properties** directly from the percept-derived information. We have already seen two diagnostic rules:

    $(\forall\ l,s)\ At(Agent,l,s) \wedge Breeze(s) \rightarrow Breezy(l)$

    $(\forall\ l,s)\ At(Agent,l,s) \wedge Stench(s) \rightarrow Smelly(l)$

Representing change: The frame problem

Frame axioms: If property x doesn't change as a result of applying action a in state s, then it stays the same.

- $On\ (x, z, s) \wedge Clear\ (x, s) \rightarrow$

  $On\ (x, table, Result(Move(x, table), s)) \wedge$

  $\neg On(x, z, Result\ (Move\ (x, table), s))$
- $On\ (y, z, s) \wedge y \neq x \rightarrow On\ (y, z, Result\ (Move\ (x, table), s))$
- The proliferation of frame axioms becomes very cumbersome in complex domains

The frame problem II

- **Successor-state axiom**: General statement that characterizes every way in which a particular predicate can become true:
  - Either it can be **made true**, or it can **already be true and not be changed**:
  - $On\ (x, table, Result(a,s)) \leftrightarrow$

    $[On\ (x, z, s) \wedge Clear\ (x, s) \wedge a = Move(x, table)] \wedge$

    $[On\ (x, table, s) \wedge a \neq Move\ (x, z)]$
- In complex worlds, where you want to reason about longer chains of action, even these types of axioms are too cumbersome
  - Planning systems use special-purpose inference methods to reason about the expected state of the world at any point in time during a multi-step plan

Qualification problem

- How can you possibly characterize every single effect of an action, or every single exception that might occur?
- When I put my bread into the toaster, and push the button, it will become toasted after two minutes, unless…
  - The toaster is broken, or…
  - The power is out, or…
  - I blow a fuse, or…
  - A neutron bomb explodes nearby and fries all electrical components, or…
  - A meteor strikes the earth, and the world we know it ceases to exist, or…

Ramification problem

Similarly, it's just about impossible to characterize every side effect of every action, at every possible level of detail.

When I put my bread into the toaster, and push the button, the bread will become toasted after two minutes, and…

- The crumbs that fall off the bread onto the bottom of the toaster over tray will also become toasted, and…
- Some of the aforementioned crumbs will become burnt, and…
- The outside molecules of the bread will become "toasted," and…
- The inside molecules of the bread will remain more "breadlike," and…
- The toasting process will release a small amount of humidity into the air because of evaporation, and…
- The heating elements will become a tiny fraction more likely to burn out the next time I use the toaster, and…
- The electricity meter in the house will move up slightly, and…

Knowledge engineering!

- Modeling the "right" conditions and the "right" effects at the "right" level of abstraction is very difficult
- Knowledge engineering (creating and maintaining knowledge bases for intelligent reasoning) is an entire field of investigation
- Many researchers hope that automated knowledge acquisition and machine learning tools can fill the gap:
  - Our intelligent systems should be able to **learn** about the conditions and effects, just like we do!
  - Our intelligent systems should be able to learn when to pay attention to, or reason about, certain aspects of processes, depending on the context!

Preferences among actions

- A problem with the Wumpus world knowledge base that we have built so far is that it is difficult to decide which action is best among a number of possibilities.
- For example, to decide between a forward and a grab, axioms describing when it is OK to move to a square would have to mention glitter.
- This is not modular!
- We can solve this problem by **separating facts about actions from facts about goals**. This way our **agent can be reprogrammed just by asking it to achieve different goals**.

Preferences among actions

- The first step is to describe the desirability of actions independent of each other.
- In doing this we will use a simple scale: actions can be Great, Good, Medium, Risky, or Deadly.
- Obviously, the agent should always do the best action it can find:

  $(\forall a,s)\ \mathrm{Great}(a,s) \rightarrow \mathrm{Action}(a,s)$

  $(\forall a,s)\ \mathrm{Good}(a,s) \wedge \neg(\exists b)\ \mathrm{Great}(b,s) \rightarrow \mathrm{Action}(a,s)$

  $(\forall a,s)\ \mathrm{Medium}(a,s) \wedge (\neg(\exists b)\ \mathrm{Great}(b,s) \vee \mathrm{Good}(b,s)) \rightarrow \mathrm{Action}(a,s)$

  …

Preferences among actions

- We use this action quality scale in the following way.
- Until it finds the gold, the basic strategy for our agent is:
  - Great actions include picking up the gold when found and climbing out of the cave with the gold.
  - Good actions include moving to a square that's OK and hasn't been visited yet.
  - Medium actions include moving to a square that is OK and has already been visited.
  - Risky actions include moving to a square that is not known to be deadly or OK.
  - Deadly actions are moving into a square that is known to have a pit or a Wumpus.

Goal-based agents

- Once the gold is found, it is necessary to change strategies. So now we need a new set of action values.
- We could encode this as a rule:
  - $(\forall s)\ \mathrm{Holding}(\mathrm{Gold},s) \rightarrow \mathrm{GoalLocation}([1,1]),s)$
- We must now decide how the agent will work out a sequence of actions to accomplish the goal.
- Three possible approaches are:
  - **Inference**: good versus wasteful solutions
  - **Search**: make a problem with operators and set of states
  - **Planning**: to be discussed later

Coming up next:

- Logical inference
- Knowledge representation
- Planning