Lógica y Metodos Avanzados de Razonamiento

Today: Introduction to propositional and first-order logic

Axel Polleres

axel.polleres@urjc.es

Overview:

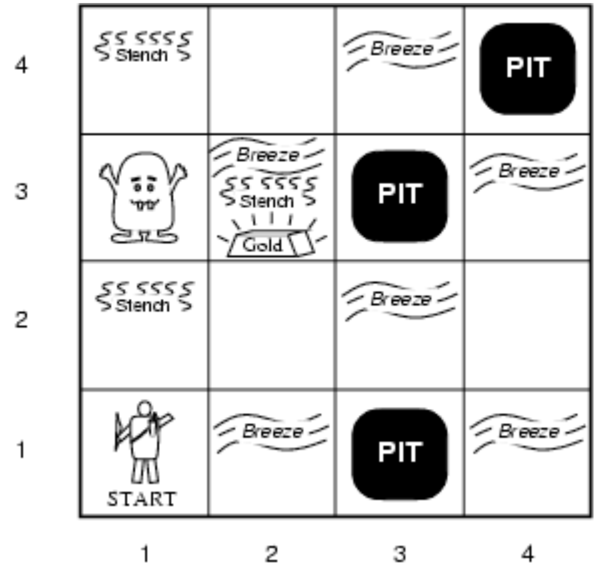
- Why logics? An example
- Propositional logics
  - Syntax, Semantics
- First-order Logics
  - Why is propositional logics not enough?
  - Syntax, Semantics
- Exercises

An example for reasoning: The "Wumpus World"

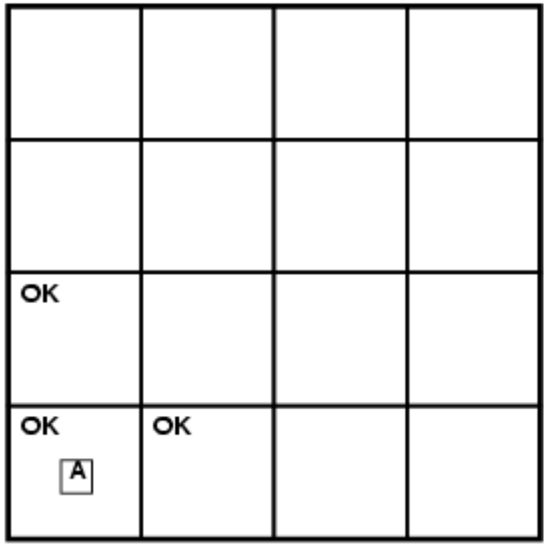
- Environment
  - Squares adjacent to wumpus are smelly (stench)
  - Squares adjacent to pit are breezy

  We want to move around in this world, without being eaten by the Wumpus or falling into pits!

- Sensors: Stench, Breeze

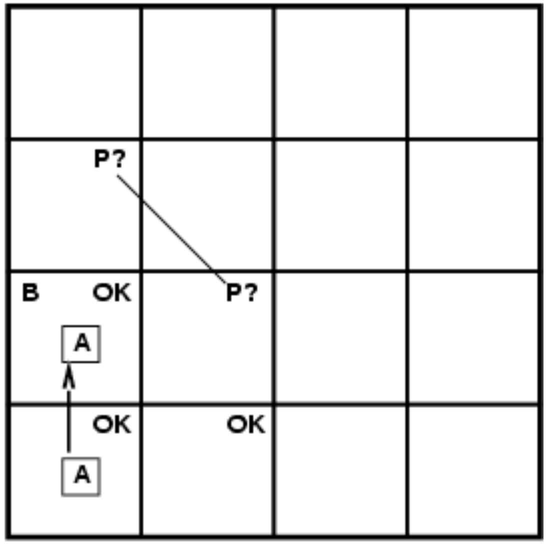


Exploring a wumpus world

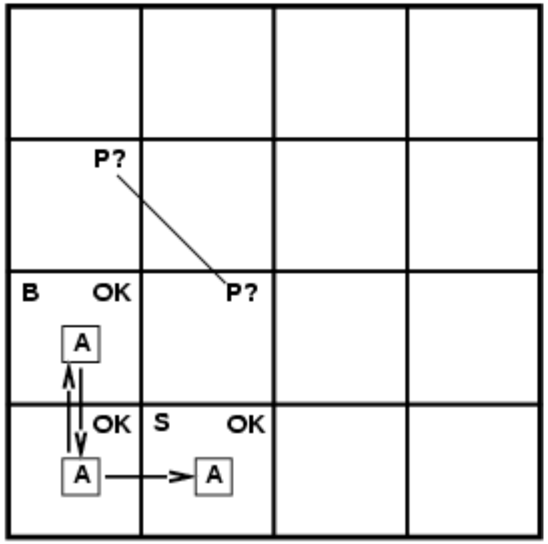


From: no stench and no breeze at [1,1] you can infer that [1,2] and [2,1] are both safe…

Exploring a wumpus world

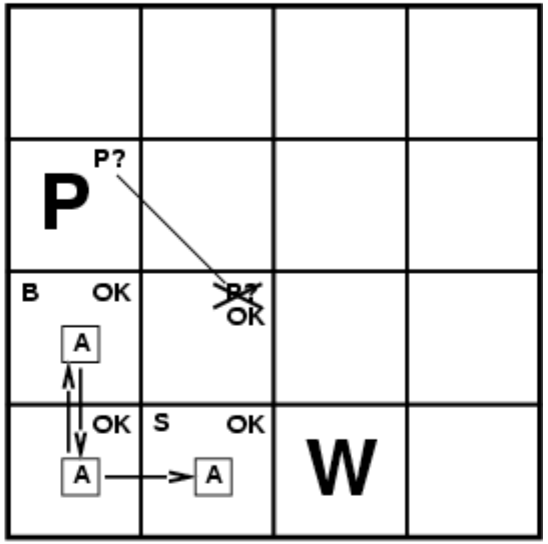


Exploring a wumpus world



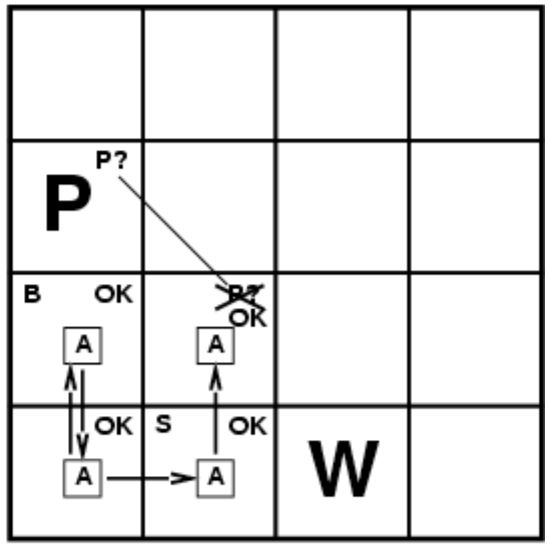
So, the only save place is to go back to [1,2]…
… but there's an awful stench…

Exploring a wumpus world

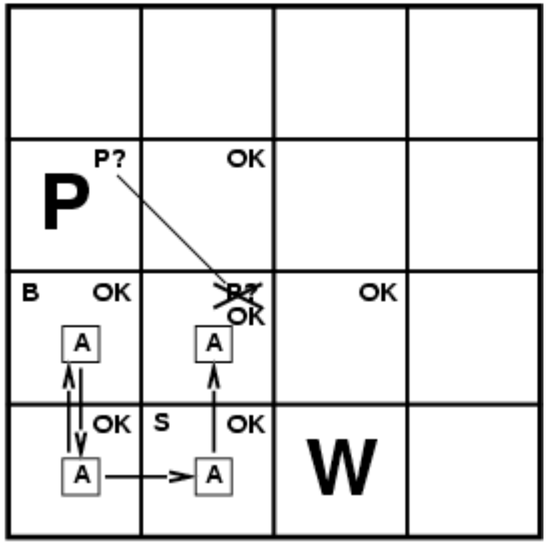


Since there's no breeze at [1,2] however, and there was no stench at [2,1]
you can infer that [2,2] is ok!

Exploring a wumpus world



Exploring a wumpus world



No breeze no stench… thus [3,2] and [2,3] both safe!

Probably you all did similar "inferences" already playing some computer games, can you program an agent playing "Minesweeper"®?

What about more tasks? E.g. a crawler exploring webpages following links according to certain rules…

Logic in general

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

- E.g., the language of arithmetic
  - $x+2 \geq y$ is a sentence; $x2+y > \{\}$ is not a sentence
  - $x+2 \geq y$ is true iff the number $x+2$ is no less than the number y
  - $x+2 \geq y$ is true in a world where $x = 7$, $y = 1$
  - $x+2 \geq y$ is false in a world where $x = 0$, $y = 6$

Entailment

- Entailment means that one thing follows from another:

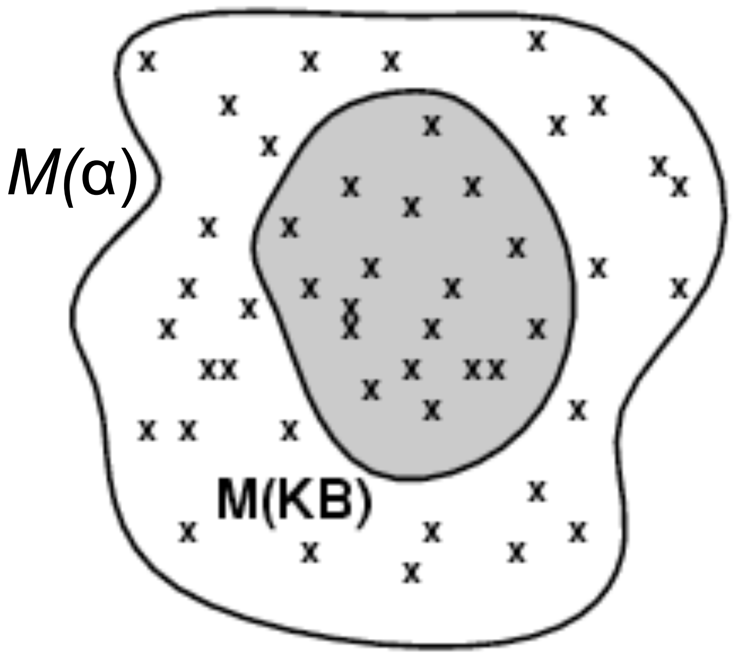
"entails"

$$KB \models \alpha$$

- Knowledge base *KB* entails sentence α if and only if α is true in all worlds where *KB* is true
  - E.g., the KB containing “It is sunny” and “It is warm” entails “It is sunny or it is warm”
  - E.g., x+y = 4 (plus basic mathematical knowledge!) entails 4 = x+y
  - Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Models

- Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated
- We say *m* is a model of a sentence α if α is true in *m*
- *M(α)* is the set of all models of α
- Then KB ⊨ α iff *M(KB)* ⊆ *M*(α)
  - E.g. *KB* = it is sunny and It is warm
  - α = it is sunny

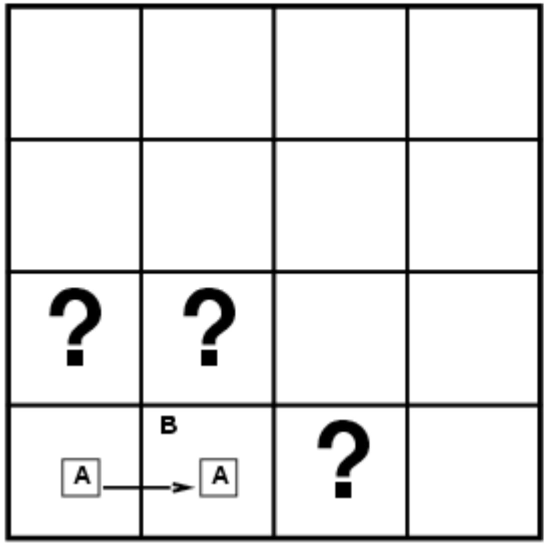


Entailment in the wumpus world

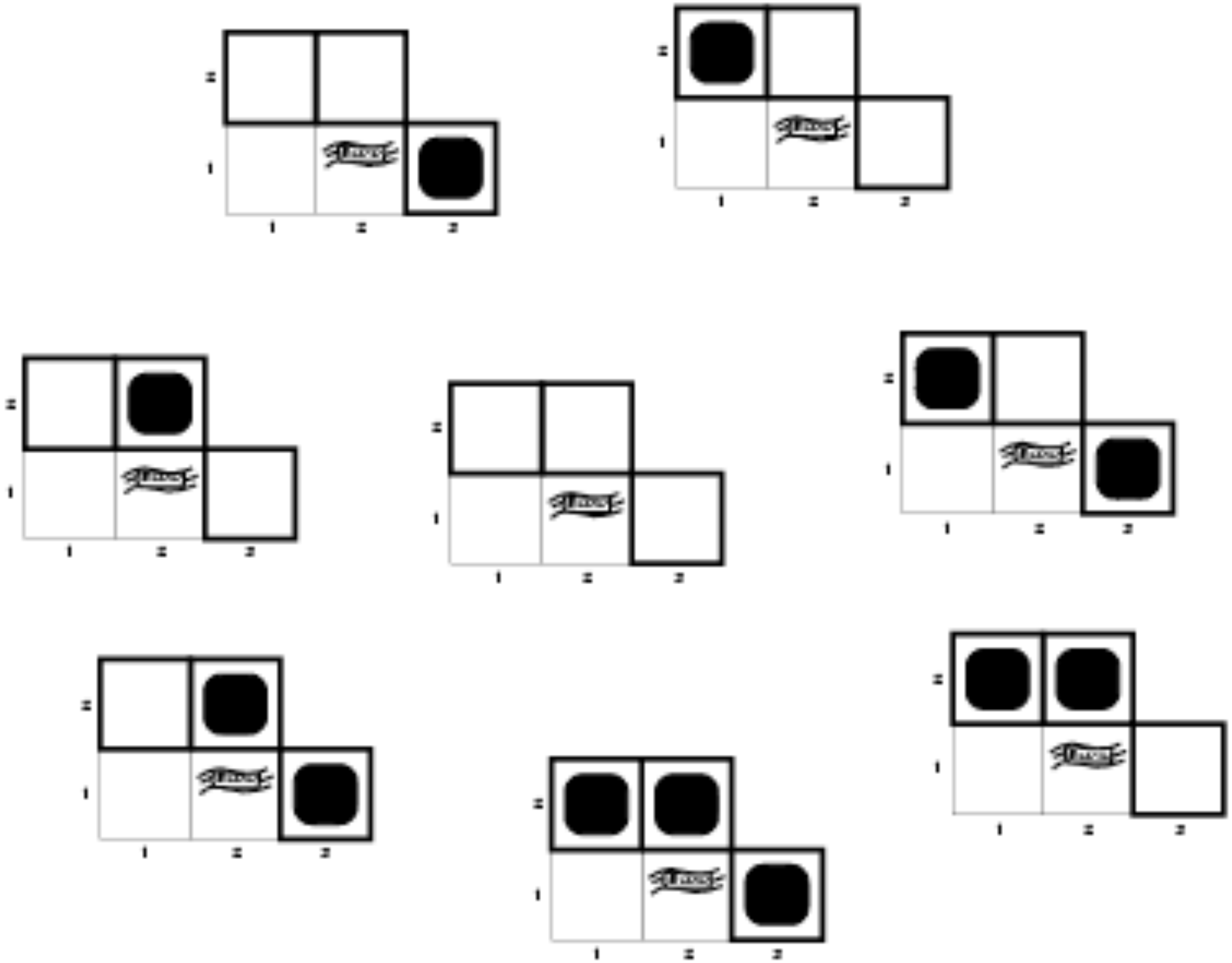
Situation after detecting nothing in [1,1], moving right, breeze in [2,1]

Consider possible models for *KB* assuming **only pits**

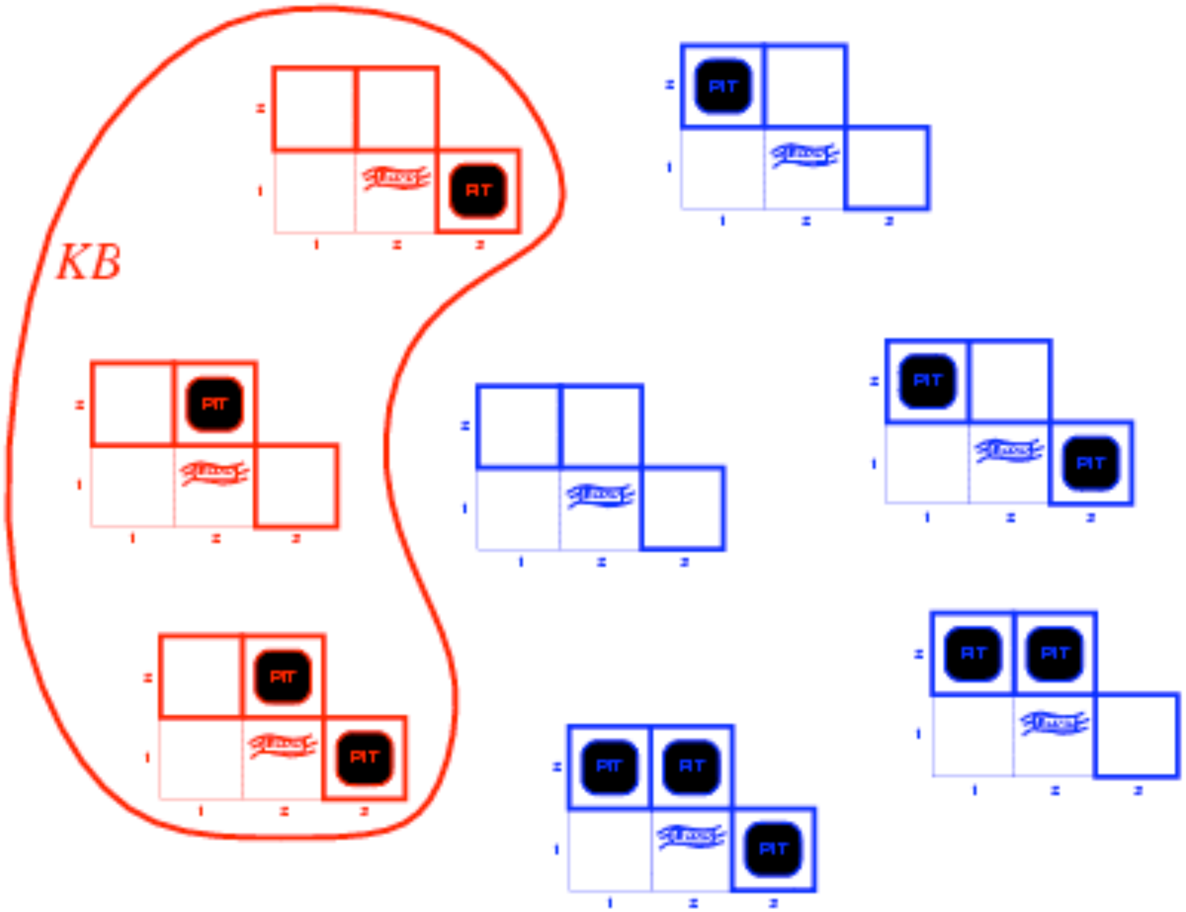
3 Boolean choices ⇒ 8 possible models (interpretations)



Wumpus models

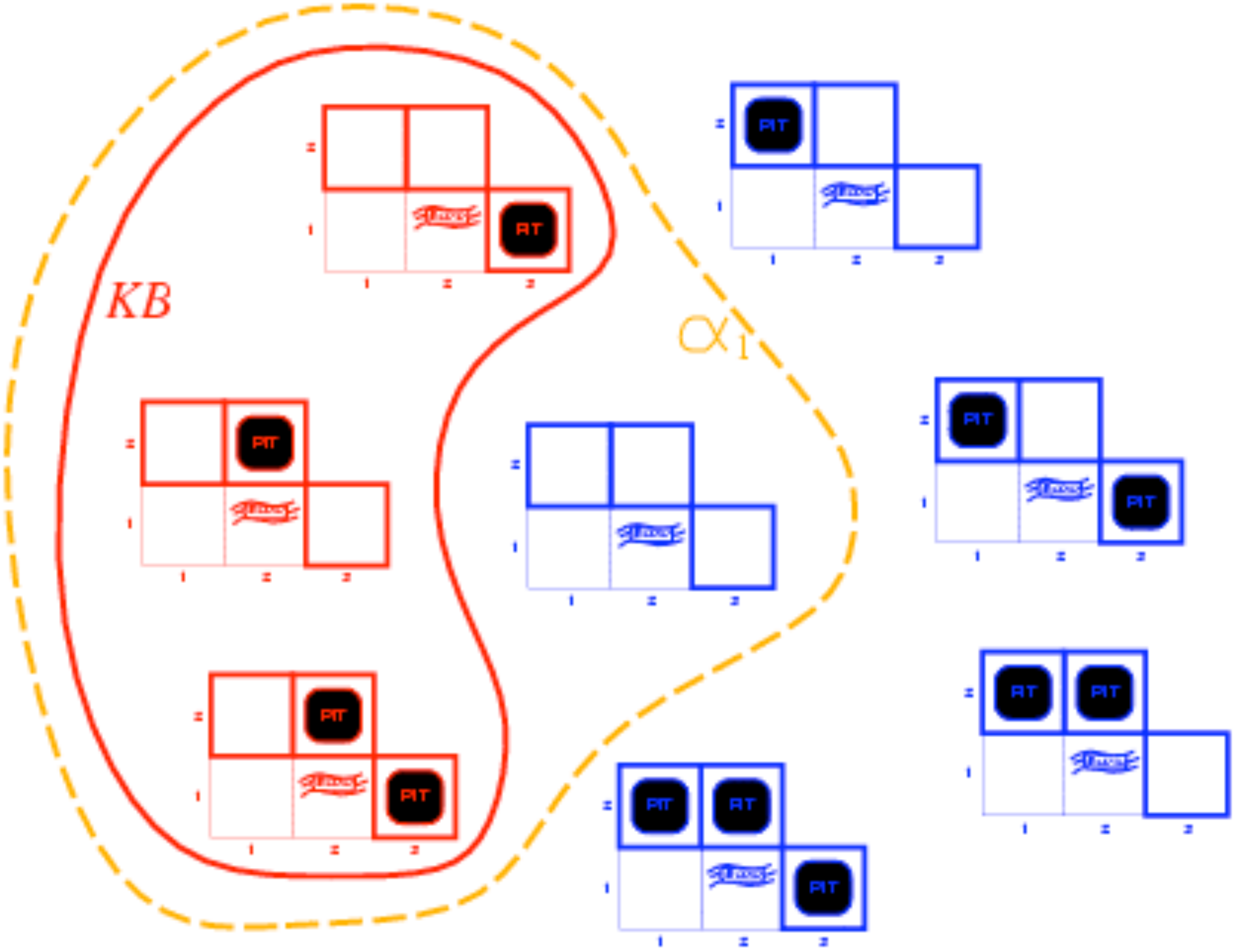


Wumpus models

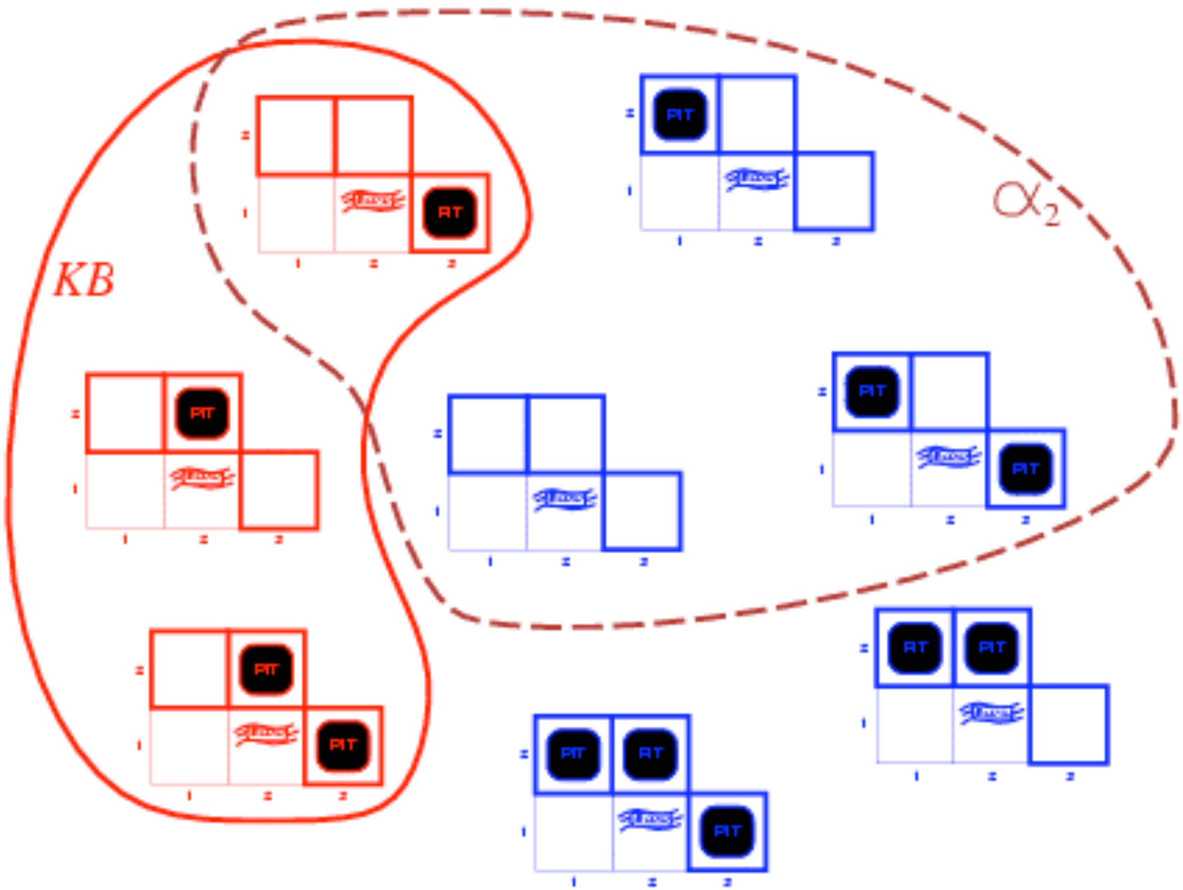


- *KB* = wumpus-world rules + observations

Wumpus models



- *KB* = wumpus-world rules + observations
- α_1 = "[1,2] is safe", *KB* $\models \alpha_1$, can be proven logically!



- *KB* = wumpus-world rules + observations
- α_2 = "[2,2] is safe", $KB \nvDash \alpha_2$

Inference

"proves"

- *KB* $\vdash_i$ α = sentence α can be derived from *KB* by procedure *i*
- Soundness: *i* is sound if whenever *KB* $\vdash_i$ α, it is also true that *KB* ⊨ α
- Completeness: *i* is complete if whenever *KB* ⊨ α, it is also true that *KB* $\vdash_i$ α
- That is, the procedure will answer any question whose answer follows from what is known by the *KB* correctly.

Propositional logic: Syntax

- Propositional logic is the simplest logic – illustrates basic ideas; its syntax is easily definable recursively as follows:

- A propositional alphabeth $\mathcal{A}$ consists of a set of proposition symbols, e.g. P_1, P_2 etc.

- Formulas are defined recursively:
  - The proposition symbols in $\mathcal{A}$ etc are sentences (aka formulae)
  - If S is a sentence, $\neg$S is a sentence (negation)
  - If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)
  - If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)
  - If S_1 and S_2 are sentences, $S_1 \rightarrow S_2$ is a sentence (implication)
  - If S_1 and S_2 are sentences, $S_1 \leftrightarrow S_2$ is a sentence (double-implication)

Propositional logic: Semantics

Each model specifies true/false for each proposition symbol

E.g.	$P_{1,2}$	$P_{2,2}$	$P_{3,1}$
	false	true	false

With these symbols, 8 possible models (interpretations)for three propositions, can be enumerated automatically.

Rules for evaluating truth with respect to an interpretation *m*:

$\neg S$	is true	iff	S is false
$S_1 \wedge S_2$	is true	iff	S_1 is true and S_2 is true
$S_1 \vee S_2$	is true	iff	S_1 is true or S_2 is true
$S_1 \rightarrow S_2$	is true	iff	S_1 is false or S_2 is true
i.e.,	is false	iff	S_1 is true and S_2 is false
$S_1 \leftrightarrow S_2$	is true	iff	$S_1 \rightarrow S_2$ is true and $S_2 \rightarrow S_1$ is true

Simple recursive process evaluates an arbitrary sentence wrt. an interpretation, e.g.,

$$\neg P_{1,2} \wedge (P_{2,2} \vee P_{3,1}) = \textit{true} \wedge (\textit{true} \vee \textit{false}) = \textit{true} \wedge \textit{true} = \textit{true}$$

Truth tables for connectives

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \rightarrow Q$	$P \leftrightarrow Q$
$false$	$false$	$true$	$false$	$false$	$true$	$true$
$false$	$true$	$true$	$false$	$true$	$true$	$false$
$true$	$false$	$false$	$false$	$true$	$false$	$false$
$true$	$true$	$false$	$true$	$true$	$true$	$true$

Wumpus world 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].

- Observations:

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

- Rules: "Pits cause breezes in adjacent squares"

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

Truth tables for inference

Remember: we want to prove: α_1 = "[1,2] is safe", i.e.,

KB = Observations $\wedge$ Rules

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

$B_{1,1}$	$B_{2,1}$	$P_{1,1}$	$P_{1,2}$	$P_{2,1}$	$P_{2,2}$	$P_{3,1}$	KB	α_1
false	*false*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*false*	*false*	*false*	*false*	*false*	*true*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
false	*true*	*false*	*false*	*false*	*false*	*false*	*false*	*true*
false	*true*	*false*	*false*	*false*	*false*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*false*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*false*	*true*	*true*	<u>*true*</u>	<u>*true*</u>
false	*true*	*false*	*false*	*true*	*false*	*false*	*false*	*true*
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
true	*true*	*true*	*true*	*true*	*true*	*true*	*false*	*false*

KB $\models \alpha_1$

Extremely naïve Inference by enumeration

- Depth-first enumeration of all models is sound and complete

```
function TT-ENTAILS?(KB, α) returns true or false

   symbols ← a list of the proposition symbols in KB and α
   return TT-CHECK-ALL(KB, α, symbols, [])

--------------------------------------------------------------

function TT-CHECK-ALL(KB, α, symbols, model) returns true or false
   if EMPTY?(symbols) then
      if PL-TRUE?(KB, model) then return PL-TRUE?(α, model)
      else return true
   else do
      P ← FIRST(symbols); rest ← REST(symbols)
      return TT-CHECK-ALL(KB, α, rest, EXTEND(P, true, model)) and
             TT-CHECK-ALL(KB, α, rest, EXTEND(P, false, model))
```

  - PL-TRUE evaluates a sentence recursively wrt. to an interpretation, see slide 25.
  - EXTEND(s,v,m) extends the partial model m by assigning value v to symbol s.

- For *n* symbols, time complexity is $O(2^n)$, space complexity is $O(n)$

Logical equivalence

- Two sentences are logically equivalent iff they are true in same models: α ≡ ß iff α ⊨ β and β ⊨ α

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

Validity and satisfiability

A sentence is valid if it is true in all models,

e.g., *True*, $A \lor \neg A$, $A \to A$, $(A \land (A \to B)) \to B$

Validity is connected to Entailment via the Deduction Theorem:

$KB \vDash \alpha$ if and only if $(KB \to \alpha)$ is valid, often written $\vDash (KB \to \alpha)$

A sentence is satisfiable if it is true in some model

e.g., $A \lor B$, C

A sentence is unsatisfiable if it is true in no models

e.g., $A \land \neg A$

Satisfiability is connected to Entailment as follows:

$KB \vDash \alpha$ if and only if $(KB \land \neg\alpha)$ is unsatisfiable

Proof methods

- We already learned a naïve proof method for propositional logic!
- In the course of this lecture we will learn more different logics and different proof methods!

- Proof methods (for propositional logics) divide into (roughly) two kinds:

  - Application of inference rules
    - Legitimate (sound) generation of new sentences from old
    - Proof = a sequence of inference rule applications
      Can use inference rules as operators in a *standard search algorithm*
    - Often require transformation of sentences into a normal form

  - Model checking
    - Truth table enumeration (always exponential in *n*)
    - Other methods, improved backtracking, e.g., Davis-Putnam-Logemann-Loveland (DPLL)
    - heuristic search in model space (sound but incomplete)
      e.g., min-conflicts-like hill-climbing algorithms

What we learned so far?

- How to write down knowledge as a propositional logical theory (Syntax)
- What does a logical theory mean (Semantics)
- How can we proof entailment naively

From propositional logic to first-order logic:

- Propositional logic has very limited expressive power
  - (unlike natural language)
  - E.g., cannot say "pits cause breezes in adjacent squares“
    - except by writing one sentence for each square
- Whereas propositional logic assumes the world contains facts (=propositional symbols),
- first-order logic (like natural language) assumes the world contains
  - Objects (constant symbols): people, houses, numbers, colors, baseball games, wars, …
  - Relations (predicate symbols): red, round, prime, brother of, bigger than, part of, comes between, …
  - Functions (function symbols): father of, best friend, one more than, plus, …

Syntax of FOL: Basic elements

- Constants KingJohn, 2, NUS,...
- Predicate symbols Brother, >,...
- Function symbols Sqrt, LeftLegOf,...
- Variables x, y, a, b,...
- Connectives $\neg$, $\rightarrow$, $\wedge$, $\vee$, $\leftrightarrow$
- Equality =
- Quantifiers $\forall$, $\exists$

Atomic sentences

Atomic sentences: *predicate* ($term_1$,...,$term_n$)
or $term_1 = term_2$

Terms: *function* ($term_1$,...,$term_n$)
or *constant* or *variable*

Remark: equality is a special Predicate with a fixed semantics!

- E.g., *Brother(KingJohn,RichardTheLionheart) > (Length(LeftLegOf(Richard)), Length(LeftLegOf(KingJohn)))*

Complex sentences

- Again, as in propositional logics, complex sentences are made from atomic sentences using connectives

$$\neg S,\ S_1 \wedge S_2,\ S_1 \vee S_2,\ S_1 \rightarrow S_2,\ S_1 \leftrightarrow S_2,$$

E.g. *Sibling(KingJohn,Richard)* $\rightarrow$ *Sibling(Richard,KingJohn)*

$>(1,2) \vee \leq (1,2)$

$>(1,2) \wedge \neg >(1,2)$

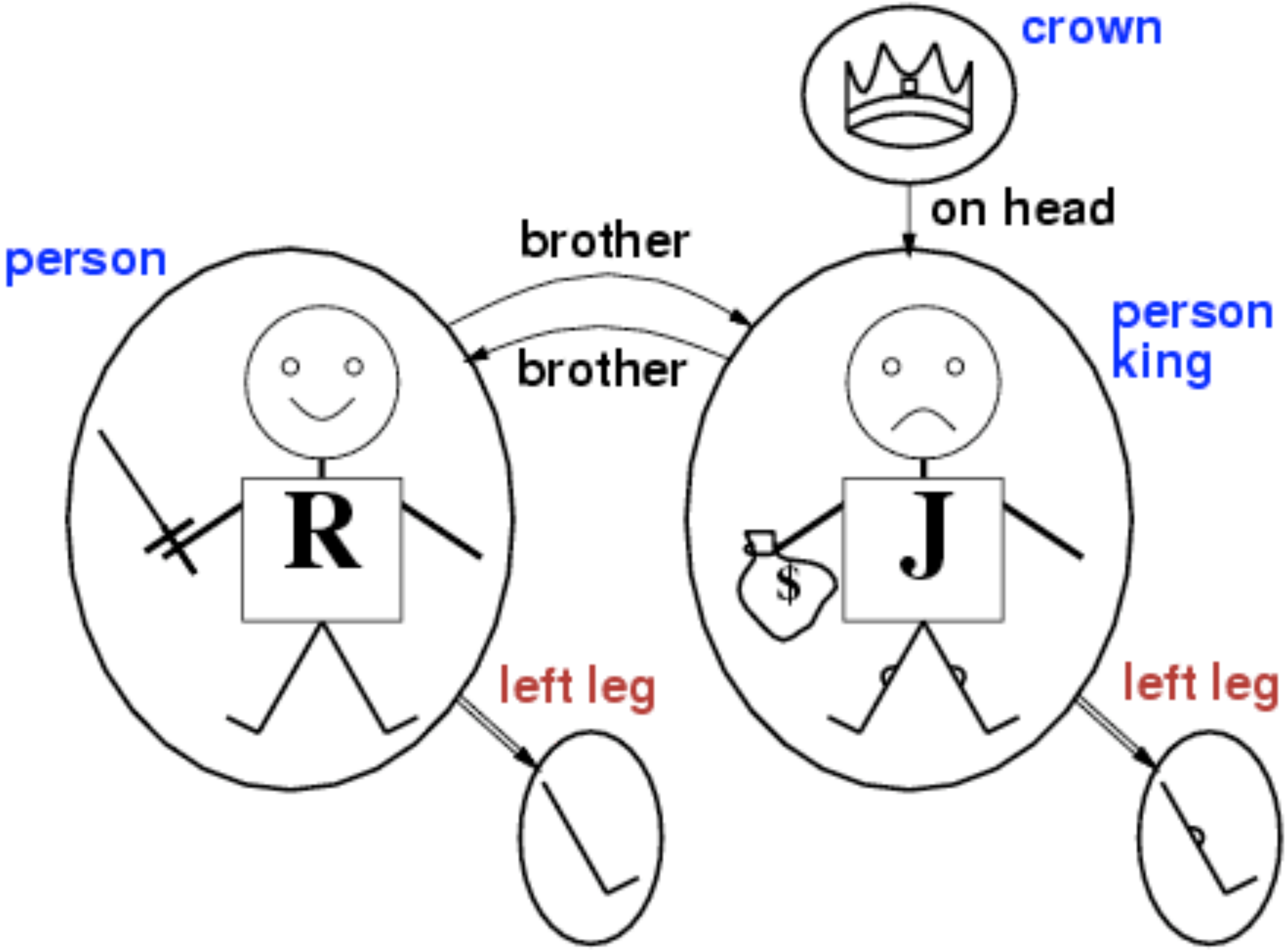
Truth in first-order logic

- Sentences are true with respect to a model and an interpretation
- Model contains objects (domain elements) and relations among them
- Interpretation specifies referents for

constant symbols	⇒	objects
predicate symbols	⇒	relations
function symbols	⇒	functions

- An atomic sentence $predicate(term_1,\ldots,term_n)$ is true iff the objects referred to by $term_1,\ldots,term_n$ are in the relation referred to by *predicate*

Models for FOL: Example



Truth in the example

Consider the interpretation in which

$Richard$ ⇒ Richard the Lionheart

$John$ ⇒ the evil King John

$Brother$ ⇒ the brotherhood relation

Under this interpretation, $Brother(Richard, John)$ is true just in case Richard the Lionheart and the evil King John are in the brotherhood relation in the model

Models in FOL

Entailment in propositional logic can be computed by enumerating models

We **can** enumerate the FOL models for a given KB vocabulary:

For each number of domain elements n from 1 to ∞
    For each k-ary predicate P_k in the vocabulary
        For each possible k-ary relation on n objects
            For each constant symbol C in the vocabulary
                For each choice of referent for C from n objects . . .

Computing entailment by enumerating FOL models is not easy!

… probably not a good idea to try to enumerate models.
… you should have heard (in some previous lectures) that
FOL nonentailment is even **undecidable,** i.e. cannot be computed ☹!

Universal quantification

- ∀<*variables*> <*sentence*>

"Everyone at URJC is smart":
$\forall x\ At(x,URJC) \rightarrow Smart(x)$

- ∀x *P* is true in a model *m* iff *P* is true with *x* being **each** possible object in the model

- Roughly speaking, equivalent to the conjunction of instantiations of *P*

$$
\begin{aligned}
& At(KingJohn, URJC) \rightarrow Smart(KingJohn) \\
\wedge\ & At(Richard, URJC) \rightarrow Smart(Richard) \\
\wedge\ & At(URJC, URJC) \rightarrow Smart(URJC) \\
\wedge\ & \ldots
\end{aligned}
$$

A common mistake to avoid

- Typically, $\rightarrow$ is the main connective with $\forall$
- Common mistake: using $\wedge$ as the main connective with $\forall$:

  $\forall x\ At(x,UIBK) \wedge Smart(x)$

  means “Everyone is at UIBK **and** everyone is smart”

- Correct: $\forall x\ At(x,UIBK) \rightarrow Smart(x)$

Existential quantification

- ∃<*variables*> <*sentence*>

- "Someone at URJC is smart":
- ∃*x* At(x,URJC) ∧ Smart(x)

- ∃*x* *P* is true in a model *m* iff *P* is true with *x* being **some** possible object in the model

- Roughly speaking, equivalent to the disjunction of instantiations of *P*

  At(KingJohn,URJC) ∧ Smart(KingJohn)
  ∨ At(Richard,URJC) ∧ Smart(Richard)
  ∨ At(UIBK,URJC) ∧ Smart(UIBK)
  ∨ ...

Another common mistake to avoid

- Typically, $\wedge$ is the main connective with $\exists$
- Common mistake: using $\rightarrow$ as the main connective with $\exists$:

  $$\exists x \; At(x,URJC) \rightarrow Smart(x)$$

  is true if there is anyone who is not at URJC!

Usually used in **Queries**:

"Is there someone in URJC who is smart?"

Correct: $\exists x \; At(x,UIBK) \wedge Smart(x)$

Properties of quantifiers

- $\forall x\ \forall y$ is the same as $\forall y\ \forall x$
- $\exists x\ \exists y$ is the same as $\exists y\ \exists x$

- $\exists x\ \forall y$ is not the same as $\forall y\ \exists x$
- $\exists x\ \forall y$ Loves(x,y)
  - "There is a person who loves everyone in the world"
- $\forall y\ \exists x$ Loves(x,y)
  - "Everyone in the world is loved by at least one person"

- Quantifier duality: each can be expressed using the other
- $\forall x$ Likes(x,IceCream) $\neg \exists x\ \neg$Likes(x,IceCream)
- $\exists x$ Likes(x,Broccoli) $\neg \forall x\ \neg$Likes(x,Broccoli)

Equality

- $term_1 = term_2$ is true under a given interpretation if and only if $term_1$ and $term_2$ refer to the same object

- E.g., definition of *Sibling* in terms of *Parent*:

  ∀*x,y Sibling(x,y)* ↔ [¬(x = y) ∧ ∃m,f ¬ (m = f) ∧ Parent(m,x) ∧ Parent(f,x) ∧ Parent(m,y) ∧ Parent(f,y)]

Using FOL

The family domain:

- Brothers are siblings

  $\forall$x,y *Brother(x,y)* $\rightarrow$ *Sibling(x,y)*

- One's mother is one's female parent

  $\forall$m,c *motherOf(c)* = m $\leftrightarrow$ *(Female(m)* $\wedge$ *Parent(m,c))*

- "Sibling" is symmetric

  $\forall$x,y *Sibling(x,y)* $\leftrightarrow$ *Sibling(y,x)*

Attention! motherOf is a function symbol here, whereas Fmale and Parent are predicate symbols!!!

Now to the formal part!

- So far we only treated FOL quite informally…
- … Now let us introduce syntax and semantics formally!

First Order Logic - Syntax

First-Order Language - Signature:

constants sometimes are denoted *0-ary* function symbols

- A set of ***constants***, e.g. *axel,logíca, 1,2,3,4, ...*
- a set of ***function symbols*** , each with a fixed *arity* $\geq$ 0 e.g.
  f, g, date, motherOf

  f(x), g(x,y), date(24,3,1974)
- a set of ***predicate symbols***, each with a fixed *arity* $\geq$ 0 e.g.
  p,ok,holdsLecture, female

  p(x,f(y)), ok, holdsLecture(axel, logíca,date(18,10,2006))
- a set of **variables**, e.g.
  x,y z,...
- **connectives**: $\wedge \; \vee \; \leftarrow \; \rightarrow \; \leftrightarrow \; \neg$
- **quantifiers**: $\forall \; \exists$
- **punctuation symbols**: ( ) ,

First Order Language - Syntax: Terms

- **Terms** consist of constants, function symbols and variables:
  - a **variable** is a term
  - each **constant** (0-ary function symbol) is a term
  - if f is an n-ary **function symbol** with n>0 and $t_1, \ldots, t_n$ are terms then $f(t_1, \ldots, t_n)$ is a term.

First Order Language - Syntax: Formulas

- **Formulae** consist of predicates, punctuation symbols, quantifiers connectives:
  - if p is an n-ary predicate symbol with n$\geq$0 and $t_1, \ldots, t_n$ are terms then $p(t_1, \ldots, t_n)$ is a formula (**atomic formula**, or **atoms**)
  - if F, G are formulae, so are
    $(\neg F), (F \vee G), (F \wedge G), (F \leftarrow G), (F \rightarrow G), F \leftrightarrow G)$
  - if F is a formula and x is a variable then
    $\exists\, x\, F$ and $\forall\, x\, F$
    are formulae as well
  - atoms and there negations are also called "literals".

Precedence of connectives:

$\neg, \forall, \exists$	negation, for all, exists
$\vee$	or
$\wedge$	and
$\leftarrow, \rightarrow$	left/right implication,
$\leftrightarrow$	equivalence

Following these precedence rules, parentheses may be skipped.

Some examples… *

$\forall x\, f(x,x) \wedge g$	*no*
$\exists y\, p(x,f(x,y)) \rightarrow q(g(y))$	*yes*
$\exists x\, p(x,f(x,y)) \rightarrow q(f(y))$	*no*
$\forall x \forall y\, (anc(x,y) \wedge father(y,z) \rightarrow anc(x,z))$	*yes*
$\forall x \exists y\, \mathrm{p}(\mathrm{x},\mathrm{y})$	yes
$\exists y \forall x\, \mathrm{p}(\mathrm{x},\mathrm{y})$	yes
$\forall x \forall y\, (anc(x,y) \wedge (father(y,z) \vee mother(y,z)) \rightarrow anc(x,z))$	*yes*
$\forall x \forall y\, (add(succ(x),y,succ(z)) \leftarrow add(x,y,z))$	*yes*
$\exists x \neg\, p(x,f(x,y)) \vee q(g(y))$	*yes*
$p(f(g(x),y),f(f(x,x),x))$	*yes*
$\neg\, p(f(g(x),y),p(f(x,x),x))$	*no*
$\forall x\, (person(x) \wedge \neg\, sleeping(x) \rightarrow awake(x))$	*yes*

* Here *f,g,h,…* denote function symbols, *p,q,r,s,…* denote predicate symbols

Bounded variables, scope and closed formulae:

- For a formula

$$\forall x\, F \text{ or } \exists x\, F$$

the **scope** of x is F. Each occurrence of x in F is **bound**. Occurrences of variables out of the scope of a quantifier are called **free**.

- Examples:

$$\forall x\, ((\exists x\, q(y,f(x))) \vee p(x)) \wedge r(x)$$
$$\exists y\, p(x,f(x,y)) \rightarrow q(g(y))$$

- A formula without free variable occurrences is called **closed,**
- Closed formulas are also called sentences

- Shortcut:

$$\forall(F) \text{ (or } \exists(F)\text{,resp.)}$$

denotes the universal (or existential, resp.) closure of a formula, i.e. the formula obtained by universally/existentially quantifying all free variables in F.

Interpretations and variable assignments:

Interpratations give some meaning to function symbols and predicate symbols…

- An **interpretation** $\mathcal{I}$ consists of:
  - a domain D over which the variables can range
  - for each n-ary *function symbol* f a *mapping* f' from $D^n \to D$ (particularly each constant is assigned an element of D)
  - for each n-ary *predicate symbol* an n-ary *relation* over the domain D, i.e. a mapping from D^n to $\{true, false\}$
- A **variable assignment** $\mathcal{V}$ wrt. an interpretation $\mathcal{I}$ is an assignment of an element of D to each variable.

Truth Value of a Formula wrt. an Interpretation $\mathcal{I}$ and a variable assignment $\mathcal{V}$

- Let $\mathcal{I}$ be an interpretation and $\mathcal{V}$ a variable assignment. Then each formula W is given a truth value $\in \{true, false\}$, written $Val^{\mathcal{I},\mathcal{V}}(W)$ as follows:

(a) If W is an atomic formula $p(t_1, \ldots, t_n)$ then

$$Val^{\mathcal{I},\mathcal{V}}(p(t_1,\ldots,t_n)) = \begin{cases} true & \text{iff } p^{\mathcal{I}}(t_1^{\mathcal{I},\mathcal{V}},\ldots,t_n^{\mathcal{I},\mathcal{V}}) = true \\ false & \text{otherwise} \end{cases}$$

(b) If W is of the form

$$\neg G \text{ then} \quad Val^{\mathcal{I},\mathcal{V}}(W) = \begin{cases} true & \text{iff } Val^{\mathcal{I},\mathcal{V}}(G) = false \\ false & \text{otherwise} \end{cases}$$

$$F \wedge G \text{ then} \quad Val^{\mathcal{I},\mathcal{V}}(W) = \begin{cases} true & \text{iff } Val^{\mathcal{I},\mathcal{V}}(F) = true \text{ and } Val^{\mathcal{I},\mathcal{V}}(G) = true \\ false & \text{otherwise} \end{cases}$$

$$F \vee G \text{ then} \quad Val^{\mathcal{I},\mathcal{V}}(W) = \begin{cases} true & \text{iff } Val^{\mathcal{I},\mathcal{V}}(F) = true \text{ or } Val^{\mathcal{I},\mathcal{V}}(G) = true \\ false & \text{otherwise} \end{cases}$$

$$F \rightarrow G \text{ then} \quad Val^{\mathcal{I},\mathcal{V}}(W) = \begin{cases} true & \text{iff } Val^{\mathcal{I},\mathcal{V}}(F) = false \text{ or } Val^{\mathcal{I},\mathcal{V}}(G) = true \\ false & \text{otherwise} \end{cases}$$

$$\exists x F \text{ then} \quad Val^{\mathcal{I},\mathcal{V}}(W) = \begin{cases} true & \text{iff there exists a } d \in D \text{ with } Val^{\mathcal{I},\mathcal{V}(x/d)} = true \\ false & \text{otherwise} \end{cases}$$

$$\forall x F \text{ then} \quad Val^{\mathcal{I},\mathcal{V}}(W) = \begin{cases} true & \text{iff for all } d \in D \; Val^{\mathcal{I},\mathcal{V}(x/d)} = true \\ false & \text{otherwise} \end{cases}$$

where $\mathcal{V}(x/d)$ is $\mathcal{V}$ except that d is assigned to x

- *Remark: The truth value of a* ***closed formula*** *does not depend on $\mathcal{V}$. So, we speak of truth values wrt. an interpretation $\mathcal{I}$, i.e. $Val^{\mathcal{I}}$).*

Models for closed formulae:

- An interpretation $\mathcal{M}$ of a closed Formula F is called a model iff $Val^{\mathcal{M}}(F) = true$

- Analogously to propositional logic, a closed formula F is called:
  - satisfiable … if it has a model
  - valid … if any interpretation is a model
  - unsatisfiable … if it doesn't have a model
  - nonvalid … if there exists an interpretation which is not a model

- Logical consequence as in propositional logic: $F \vDash G$

  Read: *"every model of F is also a model of G"*

More examples

(1) $\forall x \forall y\ (anc(x,y) \wedge father(y,z) \rightarrow anc(x,z))$

(2) $\forall x \forall y\ (anc(x,y) \wedge (father(y,z) \vee mother(y,z)) \rightarrow anc(x,z))$

(1) is satisfiable but non-valid:

D = {*franz, sepp, maria, karl, uwe, anna*}

anc … *ancestor relation*

father(x,y) … *x is father of y*

mother(x,y) … *x is mother of y*

Analogously, (2) is satisfiable but non-valid

(2) $\rightarrow$ (1) is valid!

(3)
$$father(sepp, hans) \wedge\ father(hans, karl) \wedge$$
$$\forall x \forall y (\forall z\ grandpa(x,y) \leftarrow father(x,z) \wedge father(z,y)) \wedge$$
$$\forall x \neg grandpa(sepp, x)$$

is unsatisfiable!

(For the moment you have to believe this, but we'll find out how to prove this in FOL)!

Remark:

- The notion of interpretations, models, satifiability and validity can be expanded to sets of (closed) formulae (i.e. to sets of clauses) straightforwardly:
- A set of closed formulae $S = \{F_1, \ldots, F_n\}$ is then simply viewed as the conjunction $F_1 \wedge \ldots \wedge F_n$

Some books:

- Michael R A Huth and Mark D Ryan:

  Logic in Computer Science, Cambridge University Press, 2001.
- Uwe Schöning: *Logic for Computer Scientists,* Birkhäuser Verlag, 1999.
- J.W.Lloyd: *Foundations of Logic Programming, Second edition*. Springer, 1987.

Exercises: