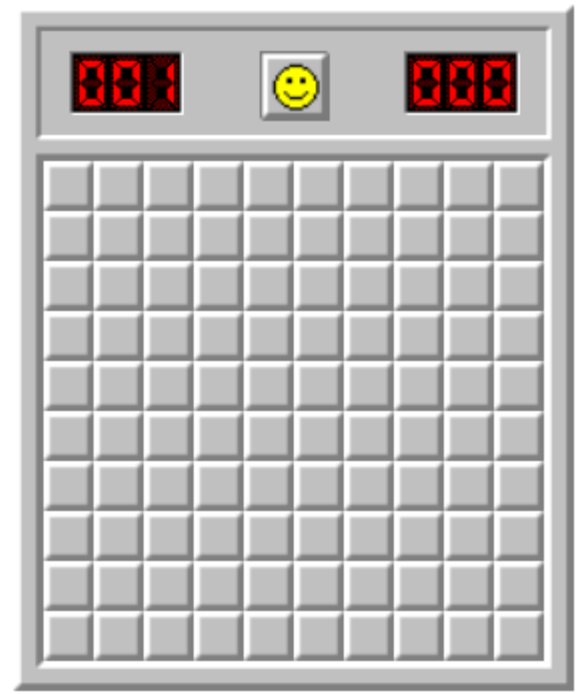
COSC76/276 Artificial Intelligence
Fall 2022
First Order Logic

Soroush Vosoughi
Computer Science
Dartmouth College
Soroush@Dartmouth.edu

Problems with propositional logic

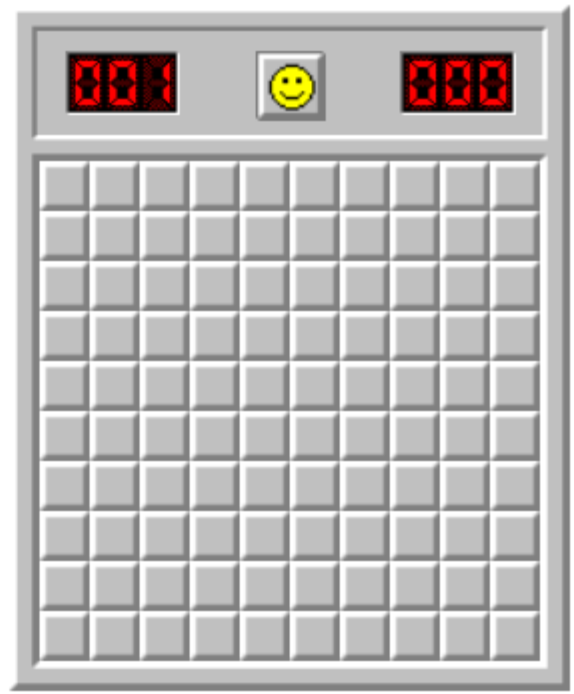
- With the game “minesweeper” on a 10x10 grid with only one landmine, how do we express in propositional logic the knowledge that squares adjacent to the landmine should display the number 1?



Discussion

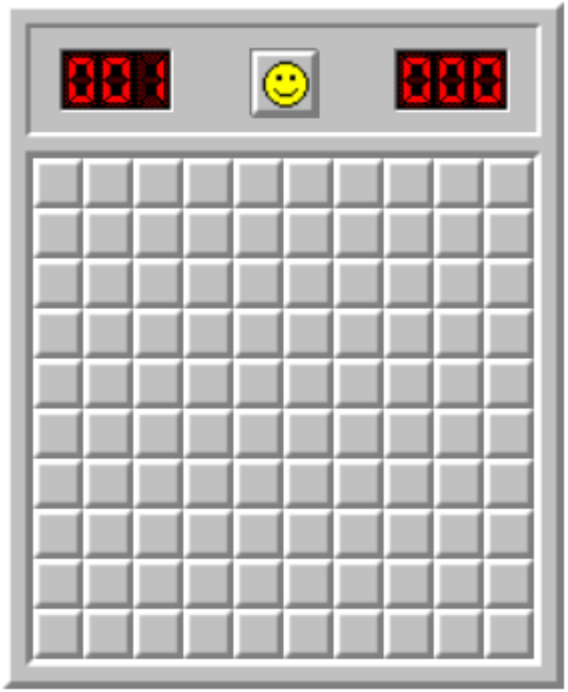
Problems with propositional logic

- For example, for cell (2,3)
  - Landmine_2_3=>number1_1_2
  - Landmine_2_3=>number1_1_3
  - Landmine_2_3=>number1_1_4
  - Landmine_2_3=>number1_2_2
  - Landmine_2_3=>number1_2_4
  - Landmine_2_3=>number1_3_2
  - Landmine_2_3=>number1_3_3
  - Landmine_2_3=>number1_3_4
- Similarly for other cells, resulting in explosion of symbols



Today's learning objectives

- We will discover the **first order logic** which allows to write
  - landmine(x,y)=>number1(neighbors(x,y))



Why not natural language?

- Can be imprecise and depend on context, e.g., "Spring":
  - mechanical?
  - a season?
  - flowing water?
- Using natural language as communication assumes large knowledge base, and seems to require some probability-based reasoning

Why not programming languages?

- Most formal languages are procedural rather than declarative. You can have objects, but don't expect Java to reason about them automatically
- There are exceptions. **Prolog** can reason about statements like "All cars are red." (Constraint satisfaction and some formal logic tools are built-in to Prolog)

Knowledge representation

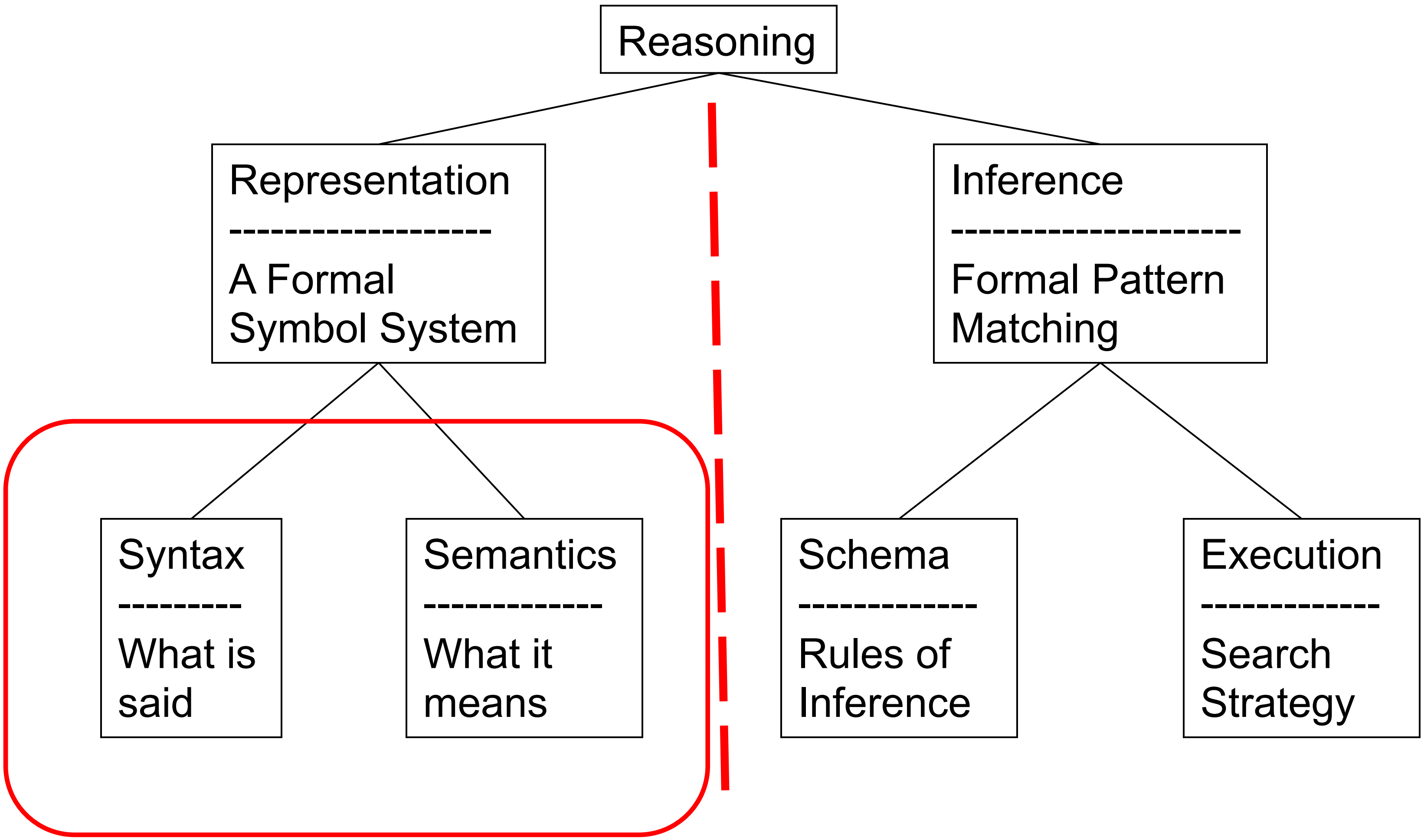
- We want something that:
  - has objects (like NL)
  - is declarative (like NL, propositional logic)
  - is context-independent (unlike NL, no "hidden" KB)
  - has a precise syntax

FOL (or FOPC) Ontology:

What kind of things exist in the world?

What do we need to describe and reason about?

Objects --- with their relations, functions, predicates, properties, and general rules.



Ontological commitment (what exists in the world)

- Looking at what propositional logic and first order logic assumes about the nature of reality

Logic	Primitives	Available knowledge
Propositional	Facts	True/false/unknown
First Order	Facts, objects, relations	True/false/unknown

First-order logic (FOL)

- First-order logic includes:
  - Objects: people, houses, numbers, …
    - Generally correspond to English nouns
  - Relations, properties, or maps take a tuple and return true or false:
    - Generally correspond to English verbs
    - First argument is generally the subject, the second the object, i.e., Verb(Noun1, Noun2) usually means “Noun1 verb noun2.”
  - Functions take in any number of objects and return one object – not true/false.

Star Wars Examples

- Objects: Leia, Luke, The Empreor, Darth Vader
- Relation (binary): siblings(Leia,Luke) -> true
- Property (a unary relation): evil(emperor) -> true
- Function: father(Luke) is DarthVader
  - or equivalently, expressed as relation,
    - father(Obiwon,Luke)=false
    - father(Leia,Luke)=false
    - father(DarthVader,Luke)=true

Syntax of FOL

- Three types of symbols:
  - Constant symbols (capture objects): KingJohn, 2, Dartmouth
  - Predicate symbols (capture relations): Brother, >,...
  - function symbols (capture functions): Sqrt, LeftLegOf
- Connectives: $\wedge \mid \vee \mid \Rightarrow \mid \Leftrightarrow \mid \neg$ (standard)
- Equality: = Two symbols refer to the same object
- Variables: x, y, z
- Quantifiers: $\forall, \exists$; ways to refer to groups of objects.

Syntax of FOL: Terms

- **Term** = logical expression that **refers to an object**
- **There are two kinds of terms:**
  - **Constant Symbols** stand for (or name) objects:
    - E.g., KingJohn, 2, UCI, Wumpus, ...
  - **Function Symbols** map tuples of objects to an object:
    - E.g., LeftLeg(KingJohn), Mother(Mary), Sqrt(x)

Syntax of FOL: Atomic Sentences

- **Atomic Sentences** state facts (logical truth values).
  - An **atomic sentence** is a Predicate symbol, followed by a parenthesized list of any argument terms
    - E.g., *Married(Father(Richard), Mother(John))*
  - An **atomic sentence** asserts that some relationship (some predicate) holds among the objects that are its arguments.
- An **Atomic Sentence is true** if the relation referred to by the predicate symbol holds among the objects (terms) referred to by the arguments.

Syntax of FOL: Atomic Sentences

- Atomic sentences in logic state facts that are true or false.

  LargerThan(2, 3) is false.

  BrotherOf(Mary, Pete) is false.

  Married(Father(Richard), Mother(John)) could be true or false.

- Note: Functions refer to objects, do not state facts:
  - Brother(Pete) refers to John (his brother) and is neither true nor false.
  - Plus(2, 3) refers to the number 5 and is neither true nor false.
- BrotherOf(Pete, Brother(Pete)) is True.

Binary relation is a truth value.

Function refers to John, an object in the world, i.e., John is Pete's brother. (Works well iff John is Pete's only brother.)

Syntax of FOL

- Three types of symbols:
  - Constant symbols (capture objects): KingJohn, 2, Dartmouth
  - Predicate symbols (capture relations): Brother, >,...
  - function symbols (capture functions): Sqrt, LeftLegOf
- Connectives: $\land \mid \lor \mid \Rightarrow \mid \Leftrightarrow \mid \neg$ (standard)
- Equality: = Two symbols refer to the same object
- Variables: x, y, z
- Quantifiers: $\forall$,$\exists$; ways to refer to groups of objects.

Syntax of FOL: Connectives & Complex Sentences

- **Complex Sentences** are formed in the same way, using the same logical connectives, as in propositional logic

- The **Logical Connectives**:
  - $\Leftrightarrow$ biconditional
  - $\Rightarrow$ implication
  - $\wedge$ and
  - $\vee$ or
  - $\neg$ negation

- **Semantics** for these logical connectives are the same as we already know from propositional logic.

Examples

- Brother(Richard, John) $\wedge$ Brother(John, Richard)
- King(Richard) $\vee$ King(John)
- King(John) => $\neg$ King(Richard)

(Semantics of complex sentences are the same as in propositional logic)

Syntax of FOL

- Three types of symbols:
  - Constant symbols (capture objects): KingJohn, 2, Dartmouth
  - Predicate symbols (capture relations): Brother, >,...
  - function symbols (capture functions): Sqrt, LeftLegOf
- Connectives: $\wedge | \vee | \Rightarrow | \Leftrightarrow | \neg$ (standard)
- Equality: = Two symbols refer to the same object
- Variables: x, y, z
- Quantifiers: $\forall, \exists$; ways to refer to groups of objects.

Syntax of FOL: Variables

- **Variables** range over objects in the world.
- A **variable** is like a **term** because it represents an object.
- A **variable** may be used wherever a **term** may be used.
  - **Variables** may be arguments to functions and predicates.
- (A **term with NO variables** is called a **ground term**.)
- (A **variable not bound by a quantifier** is called **free**.)
  - All variables we will use are bound by a quantifier.

Universal quantification

- Universal (∀)
  - Sentence is true for all values of x in the domain of variable x.
  - Conjunction of all sentences obtained by substitution of an object for the quantified variable

- $\forall x$ human(x)⇒mammal(x)
  - What it really means (universal instantiation):

    human(John)⇒mammal(John)

    (∧) human(Alice)⇒mammal(Alice)

    (∧) human(laptop)⇒mammal(laptop)

    …

Is this a correct sentence?

- $\forall x \; human(x) \wedge mammal(x)$

Discussion

Common mistake for universal quantification

- Common mistake is to use AND as main connective
  $\forall$ x human(x) $\land$ mammal(x)
  - This means everything is human and a mammal!
  - (human(Jerry) $\land$ mammal(Jerry $\land$ (human(laptop) $\land$ mammal(laptop)) $\land$ …
- **Note that => is the natural connective to use with $\forall$.**

Existential quantifiers

- Existential (∃)
  - Sentence is true for some value of x in the domain of variable x
  - Is equivalent to disjunction of all sentences obtained by substitution of an object for the quantified variable.

- "some humans are male"
  - $\exists x\ human(x) \land male(x)$
  - Means there is an x who is a human and is a male
  - What it really means (existential instantiation):

    $(human(Jerry) \land male(Jerry)) \lor$

    $(human(laptop) \land male(laptop)) \lor \ldots$

- “Some pig can fly” $\exists$ x pig(x) => fly(x) (correct?)

Common mistake for existential quantifiers

- Common mistake is to use => as main connective
- "Some pig can fly" $\exists$ x pig(x) => fly(x) (wrong)
  - This is true if there is something not a pig!
    (pig(Jerry) => fly(Jerry)) $\vee$
    (pig(laptop) => fly(laptop)) $\vee$ …
- **Note that $\wedge$ is the natural connective to use with $\exists$.**

Combining Quantifiers – Order (Scope)

The order of “like” quantifiers does not matter.

$$\forall x \, \forall y \, P(x, y) \equiv \forall y \, \forall x \, P(x, y)$$

$$\exists x \, \exists y \, P(x, y) \equiv \exists y \, \exists x \, P(x, y)$$

Like nested ANDs and ANDs in a logical sentence

Combining Quantifiers – Order (Scope)

The order of "unlike" quantifiers is important.

Like nested ANDs and ORs in a logical sentence.

$\forall$ x $\exists$ y Loves(x,y)

- For everyone ("all x") there is someone ("exists y") whom they love.
- There might be a different y for each x (y is inside the scope of x)

$\exists$ y $\forall$ x Loves(x,y)

- There is someone ("exists y") whom everyone loves ("all x").
- Every x loves the same y (x is inside the scope of y)

Parentheses can clarify: $\exists$ y ($\forall$ x Loves(x,y))

Properties of quantifiers

- $\forall x\, P(x)$ when negated becomes ?
- $\exists\, x\, P(x)$ when negated becomes ?

Properties of quantifiers

- $\forall x\ P(x)$ when negated becomes $\exists x\ \neg P(x)$
- $\exists\ x\ P(x)$ when negated becomes $\forall x\ \neg P(x)$
- Example
  - $\forall x$ sleep(x)
    - It means everybody sleeps
  - If negated, it becomes $\exists x\ \neg$sleep(x)
    - There is somebody who doesn’t sleep

Properties of quantifiers

- $\forall x\ P(x)$ is logically equivalent to $\equiv \neg\exists x\ \neg P(x)$
- $\exists\ x\ P(x)$ is logically equivalent to $\equiv \neg\forall x\ \neg P(x)$
- Example
  - $\forall x\ sleep(x)$
    - It means everybody sleeps
  - $\neg\exists x\ \neg sleep(x)$
    - There is nobody who doesn’t sleep

Connections between Quantifiers

In effect:

- $\forall$ is a conjunction over the universe of objects
- $\exists$ is a disjunction over the universe of objects

Thus, DeMorgan's rules can be applied

De Morgan's Law for Quantifiers

De Morgan's Rule

$P \wedge Q \equiv \neg(\neg P \vee \neg Q)$
$P \vee Q \equiv \neg(\neg P \wedge \neg Q)$

$\neg(P \wedge Q) \equiv (\neg P \vee \neg Q)$
$\neg(P \vee Q) \equiv (\neg P \wedge \neg Q)$

Generalized De Morgan's Rule

$\forall x\, P(x) \equiv \neg \exists x \neg P(x)$
$\exists x\, P(x) \equiv \neg \forall x \neg P(x)$

$\neg \forall x\, P(x) \equiv \exists x \neg P(x)$
$\neg \exists x\, P(x) \equiv \forall x \neg P(x)$

AND/OR Rule is simple: if you bring a negation inside a disjunction or a conjunction, always switch between them ($\neg$ OR $\rightarrow$ AND $\neg$; $\neg$ AND $\rightarrow$ OR $\neg$).

QUANTIFIER Rule is similar: if you bring a negation inside a universal or existential, always switch between them ($\neg \exists \rightarrow \forall \neg$; $\neg \forall \rightarrow \exists \neg$).

Example of sentences with quantifiers

- **“All persons are mortal.”**

  [Use: Person(x), Mortal (x)]

  $\forall x \; Person(x) \Rightarrow Mortal(x)$

- **Equivalent Forms:**

  $\forall x \; \neg Person(x) \vee Mortal(x)$

- **Common Mistakes:**

  $\forall x \; Person(x) \wedge Mortal(x)$

Example

Example of sentences with quantifiers

- **"Sissy has a sister who is a cat."**

  [Use: Sister(Sissy, x), Cat(x)]

  $\exists x\ \mathrm{Sister}(\mathrm{Sissy}, x) \wedge \mathrm{Cat}(x)$

- **Common Mistakes:**

  $\exists x\ \mathrm{Sister}(\mathrm{Sissy}, x) \Rightarrow \mathrm{Cat}(x)$

Example

Example of sentences with quantifiers

- **"For every food, there is a person who eats that food."**

  [Use: Food(x), Person(y), Eats(y, x)]

  $\forall x\ \exists y\ Food(x) \Rightarrow [\ Person(y) \wedge Eats(y, x)\ ]$

- **Equivalent Forms:**

  $\forall x\ Food(x) \Rightarrow \exists y\ [\ Person(y) \wedge Eats(y, x)\ ]$

  $\forall x\ \exists y\ \neg Food(x) \vee [\ Person(y) \wedge Eats(y, x)\ ]$

  $\forall x\ \exists y\ [\ \neg Food(x) \vee Person(y)\ ] \wedge [\neg\ Food(x) \vee Eats(y, x)\ ]$

  $\forall x\ \exists y\ [\ Food(x) \Rightarrow Person(y)\ ] \wedge [\ Food(x) \Rightarrow Eats(y, x)\ ]$

- **Common Mistakes:**

  $\forall x\ \exists y\ [\ Food(x) \wedge Person(y)\ ] \Rightarrow Eats(y, x)$

  $\forall x\ \exists y\ Food(x) \wedge Person(y) \wedge Eats(y, x)$

Example

Example of sentences with quantifiers

- **"Every person eats some food."**

  [Use: Person (x), Food (y), Eats(x, y)]

  $\forall x\ \exists y\ Person(x) \Rightarrow [\ Food(y) \wedge Eats(x, y)\ ]$

- **Equivalent Forms:**

  $\forall x\ Person(x) \Rightarrow \exists y\ [\ Food(y) \wedge Eats(x, y)\ ]$

  $\forall x\ \exists y\ \neg Person(x) \vee [\ Food(y) \wedge Eats(x, y)\ ]$

  $\forall x\ \exists y\ [\ \neg Person(x) \vee Food(y)\ ] \wedge [\ \neg Person(x) \vee Eats(x, y)\ ]$

- **Common Mistakes:**

  $\forall x\ \exists y\ [\ Person(x) \wedge Food(y)\ ] \Rightarrow Eats(x, y)$

  $\forall x\ \exists y\ Person(x) \wedge Food(y) \wedge Eats(x, y)$

Example

Example of sentences with quantifiers

- **"Some person eats some food."**

  [Use: Person (x), Food (y), Eats(x, y)]

$$\exists x\ \exists y\ \text{Person}(x) \wedge \text{Food}(y) \wedge \text{Eats}(x, y)$$

- **Common Mistakes:**

$$\exists x\ \exists y\ [\ \text{Person}(x) \wedge \text{Food}(y)\ ] \Rightarrow \text{Eats}(x, y)$$

Example

Example of sentences with quantifiers

- **“Everyone has a favorite food.”**

  [Use: Person(x), Food(y), Favorite(y, x)]

- **Equivalent Forms:**
- $\forall x\ \exists y\ \mathrm{Person}(x) \Rightarrow [\ \mathrm{Food}(y) \wedge \mathrm{Favorite}(y, x)\ ]$
- $\forall x\ \mathrm{Person}(x) \Rightarrow \exists y\ [\ \mathrm{Food}(y) \wedge \mathrm{Favorite}(y, x)\ ]$
- $\forall x\ \exists y\ \neg\mathrm{Person}(x) \vee [\ \mathrm{Food}(y) \wedge \mathrm{Favorite}(y, x)\ ]$
- $\forall x\ \exists y\ [\ \neg\mathrm{Person}(x) \vee \mathrm{Food}(y)\ ] \wedge [\ \neg\mathrm{Person}(x) \vee \mathrm{Favorite}(y, x)\ ]$
- $\forall x\ \exists y\ [\mathrm{Person}(x) \Rightarrow \mathrm{Food}(y)\ ] \wedge [\ \mathrm{Person}(x) \Rightarrow \mathrm{Favorite}(y, x)\ ]$

- **Common Mistakes:**
- $\forall x\ \exists y\ [\ \mathrm{Person}(x) \wedge \mathrm{Food}(y)\ ] \Rightarrow \mathrm{Favorite}(y, x)$
- $\forall x\ \exists y\ \mathrm{Person}(x) \wedge \mathrm{Food}(y) \wedge \mathrm{Favorite}(y, x)$

Example

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

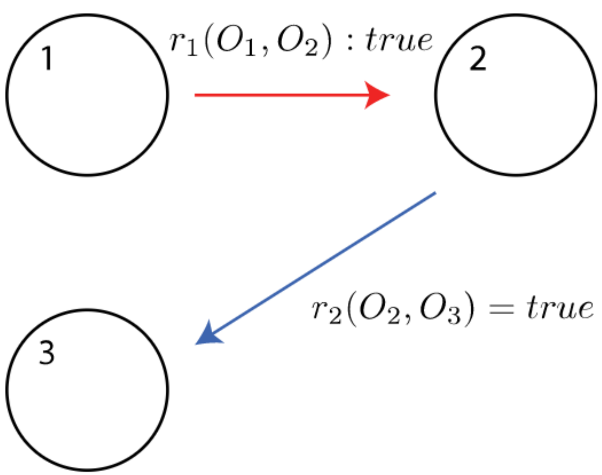
- E.g., definition of *Sibling* in terms of *Parent*, using = is:

$$
\begin{aligned}
&\forall x,y\ Sibling(x,y) \Leftrightarrow \\
&\quad [\neg(x = y) \wedge \\
&\quad \exists m,f\ \ \neg(m = f) \wedge Parent(m,x) \wedge Parent(f,x) \\
&\qquad\qquad \wedge Parent(m,y) \wedge\ Parent(f,y)]
\end{aligned}
$$

Semantics

- sentences + (model, interpretation) $\mapsto$ true/false
- interpretation specifies exactly which objects, relations, and functions are referred to by the constant, predicate, and function symbols.

  '=' sign is used
- Models, objects, relations



Models

- A set of true/false values for every relation among objects. (Think of a set of directed edges, with different colors for each relation, of graph.)

- $r_1(O_1,O_2)$=tr1(O1,O2)=t, $r_1(O_1,O_3)$=fr1(O1,O3)=f, $r_1(O_2,O_1)$=fr1(O2,O1)=f, …

How many models?

- For each binary relation (possible edge), there are n^2 possible object pairs (2-tuples), n^3 possible ternary relations, n^k possible k-ary relations.
- That's just the number of tuples. Each can be true or false. So for each relation, we get a factor of 2^(n^k) models.
- n might be infinite. (Maybe the objects are natural numbers, which can be described in FOL.)

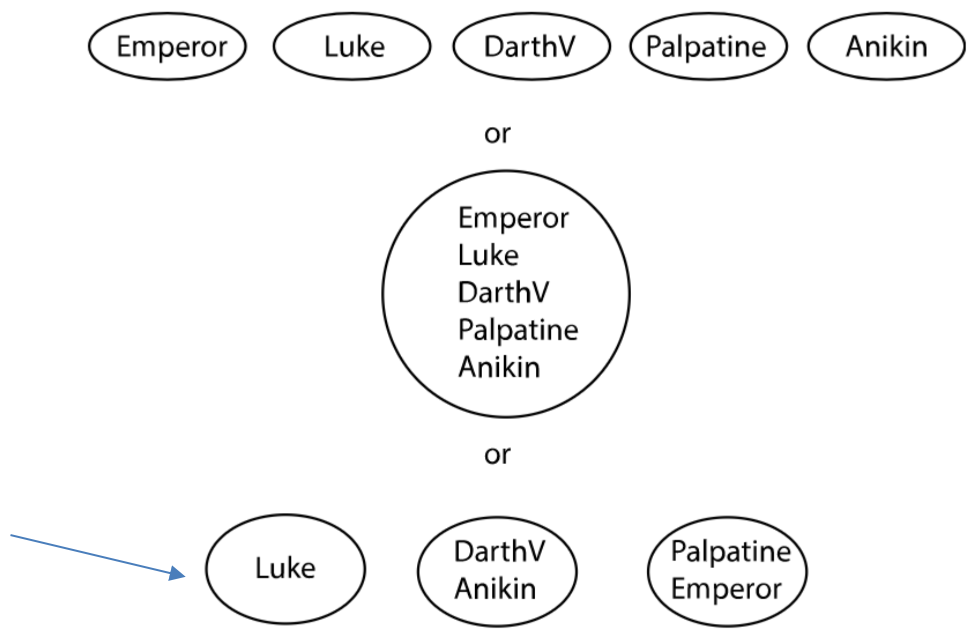
Discussion

Interpretation

- Computational complexity gets even worse
- The syntax doesn't bind symbols to particular objects

Example

- Symbols: Luke, DarthVader, Emperor, Palpatine, Anikin. Five symbols, but how many objects?



Intended interpretation

Syntactic Ambiguity

- FOL provides many ways to represent the same thing.
- E.g., "Ball-5 is red."
  - HasColor(Ball-5, Red)
    - Ball-5 and Red are objects related by HasColor.
  - Red(Ball-5)
    - Red is a unary predicate applied to the Ball-5 object.
  - HasProperty(Ball-5, Color, Red)
    - Ball-5, Color, and Red are objects related by HasProperty.
  - ColorOf(Ball-5) = Red
    - Ball-5 and Red are objects, and ColorOf() is a function.
  - HasColor(Ball-5(), Red())
    - Ball-5() and Red() are functions of zero arguments that both return an object, which objects are related by HasColor.
  - …
- This can GREATLY confuse a pattern-matching reasoner.
  - Especially if multiple people collaborate to build the KB, and they all have different representational conventions.

Syntactic Ambiguity – Partial solution

- FOL can be TOO expressive, can offer TOO MANY choices
- Likely confusion, especially for **teams** of Knowledge Engineers
- Different team members can make different representation choices
  - E.g., represent “Ball43 is Red.” as:
    - a property (= adjective)? E.g., “Red(Ball43)” ?
    - an object (= noun)? E.g., “Red = Color(Ball43))” ?
    - a predicate (= verb)? E.g., “HasProperty(Ball43, Red)” ?
- PARTIAL SOLUTION:
  - An upon-agreed **ontology** that settles these questions
  - Ontology = what exists in the world & how it is represented
  - The Knowledge Engineering teams agrees upon an ontology BEFORE they begin encoding knowledge

Summary

- First order logic to represent also objects and relations
  - Syntax includes sentences, predicate symbols, function symbols, constant symbols, variables, quantifiers
- Nested quantifiers
  - Order of unlike quantifiers matters (the outer scopes the inner)
    - Like nested ANDs and ORs
  - Order of like quantifiers does not matter
    - like nested ANDS and ANDs
- Semantics needs also interpretation

Next

- How do we make inference with FOL?