ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

IT3160E
Introduction to Artificial Intelligence

Chapter 4 – Knowledge and inference

Part 3: First-order logic

Lecturer:
Muriel VISANI

Department of Information Systems
School of Information and Communication Technology - HUST

Content of the course

- Chapter 1: Introduction
- Chapter 2: Intelligent agents
- Chapter 3: Problem Solving
  - Search algorithms, adversarial search
  - Constraint Satisfaction Problems
- Chapter 4: Knowledge and Inference
  - Knowledge representation
  - Propositional logic and first-order logic
- Chapter 5: Uncertain knowledge and reasoning
- Chapter 6: Advanced topics
  - Machine learning
  - Computer Vision

Outline

- Chapter 4 – part 1: Knowledge representation
- Chapter 4 – part 2: Propositional logic
- Chapter 4 – part 3: First-order logic
  - Definitions
  - Syntax in first-order logic
  - Semantics in first-order logic
  - Inference in first-order logic
    - Resolution
    - Robinson's resolution
  - Properties of first-order logic
  - Homework
  - Appendix: unification

Goal of this Lecture

Goal	Description of the goal or output requirement	Output division/ Level (I/T/U)
M1	Understand basic concepts and techniques of AI	1.2

First-order logic

Definitions

First Order Logic (FOL)

- **Motivation:**
  - Propositional logic is not powerful enough for many applications
    - For example, propositional logic cannot reason about natural numbers
  - In general, to reason about infinite domains or to express properties which are more abstract, first-order logic is preferred over propositional logic

First Order Logic (FOL)

- Idea: the world is made of objects
  - **Objects** are things with individual identities and **properties** to distinguish / define them
  - Objects are linked to each other by n-ary **relations** / **functions**
  - By using objects, relations/functions and properties, we can create **facts**

Recall: propositional logic *v.s.* first-order logic

- Propositional logic is the simplest type of logic
  - A **proposition** is a statement that is either true or false
  - Examples of simple sentences:
    - Hanoi is located in Vietnam
    - It is raining today
  - Examples of more complex sentences:
    - It is raining outside and the traffic in Hanoi is heavy.

Recall: propositional logic *v.s.* first-order logic

- First order logic is more complex than propositional logic:
  - Objects, relations, properties are explicit
- Examples of simple sentences:
  - Red(car12)
  - Brother(Peter, John)
- Examples of more complex sentences:
  - $\forall x,y$ parent$(x,y) \Rightarrow$ child(y,x)
    - Parent, child: examples of **functional** relations

First Order Logic (FOL)

- First Order Logic is about
  - **Objects**
    - Examples: people, houses, numbers, theories, Ronald McDonald, colors, baseball games, wars, centuries . . .
  - **Predicates**
    - Involve objects and their relations / functions; are either *True* or *False*
  - **Relations**
    - Can be unary relations (properties) such as: red, round, prime. . .
    - or more general n-ary relations such as: brother of, bigger than, inside, part of, has color, occurred after, owns, comes between, . . .
    - A relation takes objects as arguments and generates a predicate which can either be True or False
      - Examples: x is the father of y, Red(car12), x occurred after y…
  - **Functions**
    - Semantically, a function takes object(s) as arguments and generates exactly one object
      - Examples: father of x, best friend of x, sinus of x, beginning of year x. . .

First Order Logic (FOL)

- Examples of **predicates** (assertions, sentences):
  - "Squares neighboring the trash are smelly."
    - Objects: trash, squares
    - Property: smelly
    - Relation: neighboring
  - "Evil King John ruled England in 1200."
    - Objects: John, England, 1200
    - Relation: ruled
    - Properties: evil, king
  - "One plus two equals three."
    - Objects: one, two, three, one plus two
      - *"One plus two" is the object obtained by applying the function "plus" to the objects "one" and "two." "Three" is another name for this object*
    - Relation: equals
    - Function: plus (binary function)

First-order logic

Syntax in first-order logic

FOL Syntax

- Symbols
  - Variables: x, y, z,…
  - Constants: A, B, C, …
  - Function symbols (with arities): f, g, h, …
  - Relation symbols (with arities: define predicates): P, R, S
  - Logical connectives: (‘$\rightarrow$’, ‘$\neg$’, ‘$\leftrightarrow$’, ‘$\wedge$’, and ‘$\vee$’)
  - Grouping symbols: ‘()’, ‘[]’
  - Quantifiers: $\exists, \forall$
  - Equality: =

FOL Syntax

1. Variables, constants and function symbols are used to build **terms**
   - A, Bill, FatherOf(x), …
2. Relations and terms are used to build **predicates**
   - Tall(FatherOf(Bill)), Odd(x), Married(Tom,Marry), Loves(y,MotherOf(y)), …
3. Predicates and logical connective are used to build **sentences**
   - Even(4), $\forall$ x. Even(x) $\rightarrow$ Odd(x+1),$\exists$ x. x > 0

FOL Syntax

1. **Terms**
   - Terms are logical expressions that refer to an object
   - Variables are terms
   - Constants are terms
   - If $t_1,\ldots, t_n$ are terms and f is a **function symbol** with arity n, then $f(t_1,\ldots, t_n)$ is a term
     - For example: in English we'd say "John's left leg" rather than giving the leg a name. So, in FOL, instead of using a constant symbol, we use LeftLeg(John).

FOL Syntax

2. Predicates

- If $t_1,\ldots, t_n$ are terms and P is a relation symbol with arity n, then $P(t_1,\ldots, t_n)$ is a predicate
  - Tall(FatherOf(Bill)), Odd(X), Married(Tom,Marry), Loves(Y,MotherOf(Y)), …

FOL Syntax

3. Sentences

- True, False are sentences
- Predicates are sentences
- If α, β are sentences, then the followings are sentences

- $\exists x.\alpha$, $\forall x.\alpha$, (α), $\neg \alpha$, $\alpha \rightarrow ß$, $\alpha \leftrightarrow ß$, $\alpha \wedge ß$, $\alpha \vee ß$

FOL Formal grammar

$$
\begin{aligned}
\textit{Sentence} &\rightarrow \textit{AtomicSentence} \mid \textit{ComplexSentence} \\
\textit{AtomicSentence} &\rightarrow \textit{Predicate} \mid \textit{Predicate}(\textit{Term}, \ldots) \mid \textit{Term} = \textit{Term} \\
\textit{ComplexSentence} &\rightarrow (\ \textit{Sentence}\ ) \mid [\ \textit{Sentence}\ ] \\
&\mid \neg\ \textit{Sentence} \\
&\mid \textit{Sentence} \wedge \textit{Sentence} \\
&\mid \textit{Sentence} \vee \textit{Sentence} \\
&\mid \textit{Sentence} \Rightarrow \textit{Sentence} \\
&\mid \textit{Sentence} \Leftrightarrow \textit{Sentence} \\
&\mid \textit{Quantifier}\ \textit{Variable}, \ldots\ \textit{Sentence} \\
\\
\textit{Term} &\rightarrow \textit{Function}(\textit{Term}, \ldots) \\
&\mid \textit{Constant} \\
&\mid \textit{Variable} \\
\\
\textit{Quantifier} &\rightarrow \forall \mid \exists \\
\textit{Constant} &\rightarrow A \mid X_1 \mid \textit{John} \mid \cdots \\
\textit{Variable} &\rightarrow a \mid x \mid s \mid \cdots \\
\textit{Predicate} &\rightarrow \textit{True} \mid \textit{False} \mid \textit{After} \mid \textit{Loves} \mid \textit{Raining} \mid \cdots \\
\textit{Function} &\rightarrow \textit{Mother} \mid \textit{LeftLeg} \mid \cdots \\
\\
\text{OPERATOR PRECEDENCE} &: \neg, =, \wedge, \vee, \Rightarrow, \Leftrightarrow
\end{aligned}
$$

Exercise 1: Aristotelian forms

- Express, in FOL, the following 4 statements:
  - All Ps are Qs (where P, Q are two unary relations)
    - Solution:
  - No Ps are Qs
    - Solution:
  - Some Ps are Qs
    - Solution:
    - Remark: same sentence as “some Qs are Ps”
  - Some Ps aren’t Q’s
    - Solution:

Exercise 2: Aristotelian forms + predicates

- Let's consider now that we also have the following predicates:
  - *Orange(x)*
  - *Cat(x)*
  - *Fluffy(x)*

Exercise 2: Aristotelian forms + predicates

- Give the translation of « every orange cat is fluffy » in first-order logic
  - Solution:

$$\forall x.\ (x \text{ is an orange cat} \rightarrow x \text{ is fluffy})$$

$$\forall x.\ (x \text{ is an orange cat} \rightarrow Fluffy(x))$$

⚠ ~~$\forall x.\ (Orange(Cat(x)) \rightarrow Fluffy(x))$~~

Illegal statement

$$\forall x.\ (x \text{ is orange and } x \text{ is a cat} \rightarrow Fluffy(x))$$

$$\forall x.\ (Orange(x) \wedge Cat(x) \rightarrow Fluffy(x))$$

Exercise 3: Loving corgi

- Give the translation of « There's a corgi that loves everyone » in FOL
  - Knowing that we have the following predicates
    - *Corgi*(*x*) ; *Person*(*x*) ; *Loves*(*x*, *y*)
  - Solution:

$$\exists x.\ (x \text{ is a corgi} \wedge x \text{ loves everyone})$$

$$\exists x.\ (Corgi(x) \wedge x \text{ loves every person } y)$$

$$\exists x.\ (Corgi(x) \wedge \text{every person } y \text{ is loved by } x)$$

$$\exists x.\ (Corgi(x) \wedge \forall y.\ (Person(y) \rightarrow Loves(x, y)))$$

Exercise 4: Loved corgis

- Give the translation of « Everybody loves at least one corgi » in FOL
  - Knowing that we have the following predicates
    - *Corgi*(*x*) ; *Person*(*x*) ; *Loves*(*x*, *y*)
  - Solution:

$$\forall x.\ (Person(x) \rightarrow \textit{x loves at least one corgi})$$

$$\forall x.\ (Person(x) \rightarrow \textit{there is a corgi y that is loved by x})$$

$$\forall x.\ (Person(x) \rightarrow \exists y.\ (Corgi(y) \wedge Loves(x, y)))$$

N.B.

- *There's a corgi that loves everyone*

$$\exists x.\ (Corgi(x) \wedge \forall y.\ (Person(y) \rightarrow Loves(x, y)))$$

- *Everybody loves at least one corgi*

$$\forall x.\ (Person(x) \rightarrow \exists y.\ (Corgi(y) \wedge Loves(x, y)))$$

- Conclusion: the order of the quantifiers MATTER!!
  - ∃∀ or ∀∃
- **SO, UNLESS YOU'RE AN EXPERT, DON'T TRY TO TRANSLATE THE STATEMENT IN A SINGLE PASS**
  - Instead, start off with the original statement and **incrementally** translate it top-down, only adding in the quantifiers when you need them

Exercise 5: Yummy pancakes

- Give the translation of « Any two pancakes taste similar » in FOL
  - Knowing that we have the following predicates
    - *Pancake(x) TasteSimilar(x, y)*
  - Solution 1:

Any pancake x tastes similar to any pancake y

$$\forall x.\ (Pancake(x) \rightarrow$$
x tastes similar to any pancake y
$$)$$

$$\forall x.\ (Pancake(x) \rightarrow$$
$$\forall y.\ (Pancake(y) \rightarrow$$
x tastes similar to y
$$)$$
$$)$$

$$\forall x.\ (Pancake(x) \rightarrow$$
any pancake y tastes similar to x
$$)$$

$$\forall x.\ (Pancake(x) \rightarrow$$
$$\forall y.\ (Pancake(y) \rightarrow$$
$$TasteSimilar(x, y)$$
$$)$$
$$)$$

Exercise 5: Yummy pancakes

- Give the translation of « Any two pancakes taste similar » in FOL
  - Knowing that we have the following predicates
    - *Pancake(x) TasteSimilar(x, y)*
  - Solution 2:

Any two pancakes x and y taste similar

$\forall x.\ \forall y.\ ($*x and y are pancakes* $\rightarrow$ *x and y taste similar* $)$

$\forall x.\ \forall y.\ (Pancake(x) \wedge Pancake(y) \rightarrow TasteSimilar(x, y))$

Exercise 5: Yummy pancakes

- **N.B.** Solution 1 and Solution 2 are logically equivalent

$$\forall x.\ (Pancake(x) \rightarrow \forall y.\ (Pancake(y) \rightarrow TasteSimilar(x, y)))$$

$$\forall x.\ \forall y.\ (Pancake(x) \land Pancake(y) \rightarrow TasteSimilar(x, y))$$

$$A \rightarrow B \rightarrow C \quad \equiv \quad A \land B \rightarrow C$$

- This pattern – changing a chain of implications into a single implication with ANDs and *vice-versa* – is sometimes called **currying** and has applications in functional programming

Exercise 6: Social networks

- Give the translation of « Everyone knows at least two people » in FOL
  - Knowing that we have the following predicates
    - *Person*(*x*) *Knows*(*x*, *y*)
  - Solution 1:

Every person x knows at least two people y and z

$$\forall x.\ (Person(x) \rightarrow$$
there is a person y that x knows and a different person z that x knows.
$$)$$

$$\forall x.\ (Person(x) \rightarrow \exists y.\ (Person(y) \wedge Knows(x, y) \wedge \exists z.\ (Person(z) \wedge Knows(x, z) \wedge$$
z is a different person from y
$$)\ )\ )$$

$$\forall x.\ (Person(x) \rightarrow \exists y.\ (Person(y) \wedge Knows(x, y) \wedge$$
there is a different person z that x knows
$$)\ )$$

$$\forall x.\ (Person(x) \rightarrow \exists y.\ (Person(y) \wedge Knows(x, y) \wedge \exists z.\ (Person(z) \wedge Knows(x, z) \wedge z \neq y)\ )\ )$$

Exercise 6: Social networks

- Give the translation of « Everyone knows at least two people » in FOL
  - Knowing that we have the following predicates
    - *Person*(*x*) *Knows*(*x*, *y*)
  - Solution 1:
  - Solution 2:

Every person x knows at least two people y and z

$$\forall x.\ (Person(x) \rightarrow \exists y.\ (Person(y) \land Knows(x, y) \land \exists z.\ (Person(z) \land Knows(x, z) \land z \neq y)))$$

$$\forall x.\ (Person(x) \rightarrow \exists y.\ \exists z.\ (Person(y) \land Person(z) \land z \neq y \land Knows(x, y) \land Knows(x, z)))$$

Exercise 7: Natural numbers

- Give the translation of « The set of all natural numbers exists » in FOL
  - Knowing that we have the following predicates
    - $Set(x)$; $x \in y$; $Integer(x)$; $Negative(x)$
  - Difficulties:
    - We want to prove that something exists: quite different from Aristotelian forms…
    - How to translate « the set of all natural numbers » in FOL???

There is a set that is the set of all natural numbers

Exercise 7: Natural numbers

- Solution:
  - Start by re-formulating the statement to get rid of the « exists » formulation

There is a set that is the set of all natural numbers

$\exists S.\ (Set(S) \wedge$
S is the set of all natural numbers
$)$

~~$\exists S.\ (Set(S) \wedge \forall x.\ (x \in S \rightarrow Integer(x) \wedge \neg Negative(x)))$~~

It is possible to choose a set besides N that makes this sentence true (True for R): *e.g.* S={1} or S=∅
=> It is not enough to say that “All the elements in S are natural numbers”!!!

$(S \subseteq \mathbb{N})$

~~$\exists S.\ (Set(S) \wedge \forall x.\ (Integer(x) \wedge \neg Negative(x) \rightarrow x \in S))$~~

It is possible to choose a set besides N that makes this sentence true (True for R):
=> It is not enough to say that “S contains all the natural numbers”!!!

$(\mathbb{N} \subseteq S)$

Both statements are wrong, but in complementary ways!

Exercise 7: Natural numbers

- Solution:
  - Start by re-formulating the statement to get rid of the « exists » formulation

There is a set that is the set of all natural numbers

$$\exists S.\ (Set(S) \wedge S \subseteq \mathbb{N} \wedge \mathbb{N} \subseteq S)$$

$$\exists S.\ (Set(S) \wedge \forall x.\ (x \in S \rightarrow Integer(x) \wedge \neg Negative(x)) \wedge \forall x.\ (Integer(x) \wedge \neg Negative(x) \rightarrow x \in S))$$

$$\exists S.\ (Set(S) \wedge \forall x.\ (x \in S \leftrightarrow Integer(x) \wedge \neg Negative(x)))$$

First-order logic

Semantics in first-order logic

FOL Semantic

- **Variables**
  - Objects
- **Constants**
  - Entities (*e.g.* John)
- **Function symbol**
  - Function from object(s) to exactly 1 object
- **Relation symbol**
  - Relation between object(s); produce predicates
- **Quantifiers**
  - $\exists x.P$ true if P is true under some value (interpretation) of x
  - $\forall x.P$ true if P is true under every value (interpretation) of x
- **Logical connectives**
  - Similar to Propositional Logic, with in addition “equals”: “=”

FOL Semantic

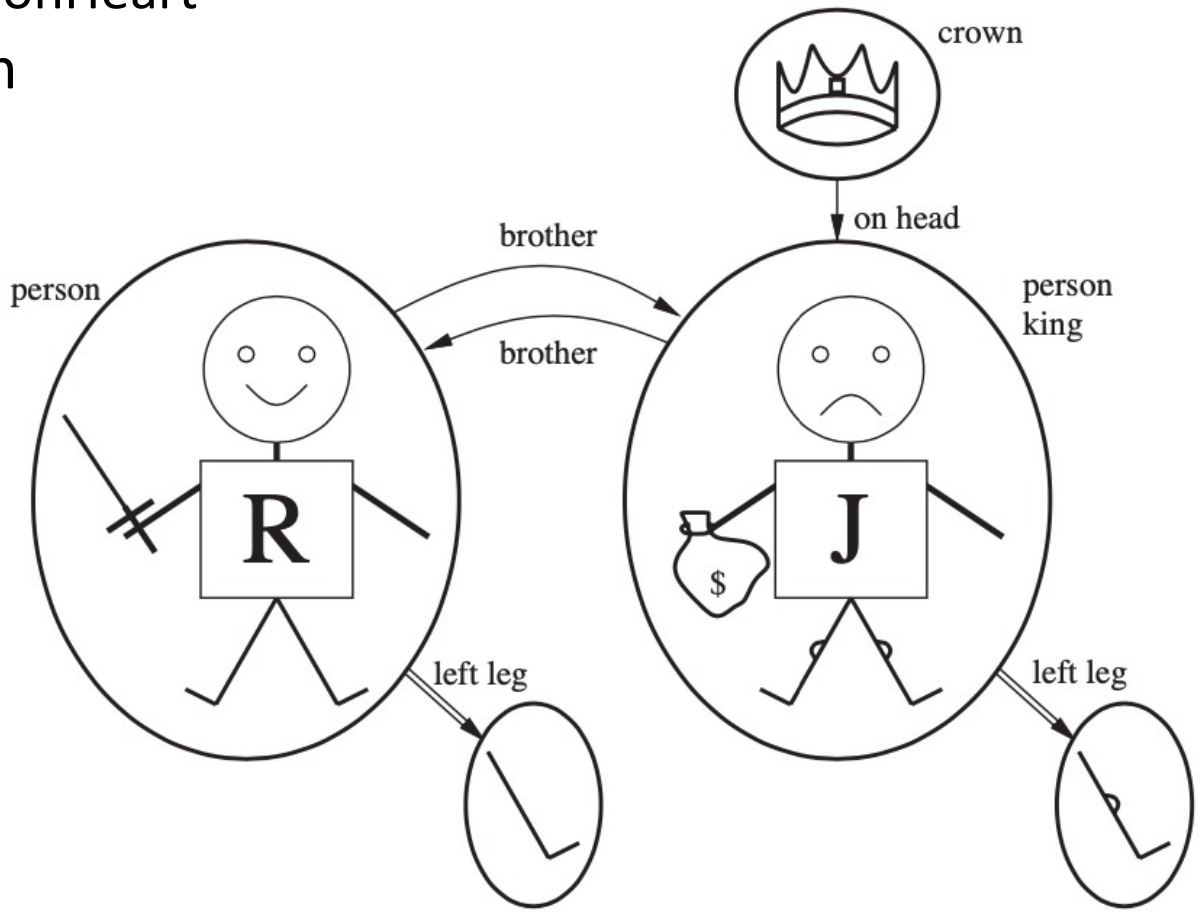
- **Interpretation:** (D, σ)
  - An interpretation specifies exactly which objects, relations and functions are referred to by the constant, predicate, and function symbols
  - D is a set of objects, called ***domain*** or *universe*
    - By definition, non-empty
  - σ is a **mapping** from variables to D
  - C^D is a member of D for each constant C
  - f^D is a mapping from D^n to D for each function symbol f with arity n
  - R^D is a relation over D^n for each relation symbol R with arity n

Example 1: simple arithmetics

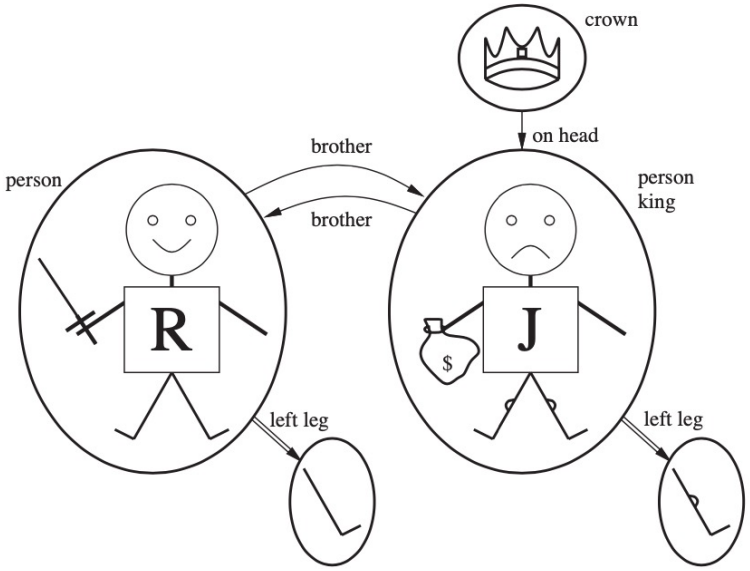
- Symbols
  - Variables: x,y,z, …
  - Constants: 0,1,2, …
  - Function symbols: +,*
  - Relation symbols: >, =
- Semantic
  - Universe: N (natural numbers)
  - The meaning of symbols
    - Constants: the meaning of 0 is *the number zero, …*
    - Function symbols: the meaning of + is *the natural number addition*, …
    - Relation symbols: the meaning of > is *the relation greater than*, …

Example 2: Kingship

- Example:
  - R: Richard the LionHeart
  - J: evil King John



Example 2: Kingship



- Questions:
  - How many objects?
    - Solution:
  - Name these Objects
    - Solution:
  - How many binary relations? Name them.
    - Solution:
  - How many unary relations?
    - Solution:
  - How many functions?
    - Solution:

FOL Semantic

- Example of (intended) interpretation (D, σ):
  - Recall: D is a set of objects, called ***domain*** or *universe*
    - Here, D: Richard the Lion Heart, evil King John, LeftLeg(John), LeftLeg(Richard), Crown
  - Recall: C^D is a member of D for each constant C
    - C^D : mapping Richard, John and C (the Crown) to D.
  - Recall: R^D is a relation over D^n for each relation symbol R with arity n:
    - Mapping "Brother" (the brotherhood relation) to D^2
    - Mapping "OnHead" to D^2
    - Mapping "person", "king" and "crown" (unary relations) to D
  - Recall: f^D is a mapping from D^n to D for each function symbol f with arity n
    - f^D : Mapping "LeftLeg" to D
  - Recall: σ is a **mapping** from variables to D
    - σ : There are many possible interpretations
      - My interpretation is that Richard refers to Richard the Lion Heart, John to Evil King John, C to (John's) crown, where Richard and John are brother, etc.
      - But, most possible interpretations don't make sense (*e.g.* the one that maps Richard to the crown)
      - The KB should be able to remove them

FOL Semantic

- The interpretation can be extended using **rules:**
  - For instance: $\forall x$ King(x) $\rightarrow$ Person(x)
  - More generally, $\forall x$ P is true in a given model if P is true in all possible extended interpretations, here whatever if x is mapped to any possible object in D:

    Richard the Lionheart,
    King John,
    Richard's left leg,
    John's left leg,
    the crown.

  - Here, it is the case (see the truth-table for logical connective $\rightarrow$)

FOL Semantic

- Given an interpretation (D, σ), the semantic of a term/sentence α is denoted

$$[\alpha]_a^D$$

- Interpretation of **terms**

$$\begin{aligned}
[\![x]\!]_\sigma^{\mathcal{D}} &:= \sigma(x) \\
[\![C]\!]_\sigma^{\mathcal{D}} &:= C^{\mathcal{D}} \\
[\![F(t_1,\ldots,t_n)]\!]_\sigma^{\mathcal{D}} &:= F^{\mathcal{D}}([\![t_1]\!]_\sigma^{\mathcal{D}},\ldots,[\![t_n]\!]_\sigma^{\mathcal{D}})
\end{aligned}$$

FOL Semantic

- Interpretation of sentences

$$
\begin{array}{lcll}
[\![R(t_1,\ldots,t_n)]\!]^{\mathcal{D}}_{\sigma} & := & True & \text{iff } \langle [\![t_1]\!]^{\mathcal{D}}_{\sigma},\ldots,[\![t_n]\!]^{\mathcal{D}}_{\sigma}\rangle \in R^{\mathcal{D}} \\
[\![\neg\varphi]\!]^{\mathcal{D}}_{\sigma} & := & True/False & \text{iff } [\![\varphi]\!]^{\mathcal{D}}_{\sigma} = False/True \\
[\![\varphi_1 \vee \varphi_2]\!]^{\mathcal{D}}_{\sigma} & := & True & \text{iff } [\![\varphi_1]\!]^{\mathcal{D}}_{\sigma} = True \text{ or } [\![\varphi_2]\!]^{\mathcal{D}}_{\sigma} = True \\
[\![\exists x\,\varphi]\!]^{\mathcal{D}}_{\sigma} & := & True & \text{iff } [\![\varphi]\!]^{\mathcal{D}}_{\sigma'} = True \text{ for some } \sigma'
\end{array}
$$

$$
\begin{array}{lcl}
[\![\varphi_1 \wedge \varphi_2]\!]^{\mathcal{D}}_{\sigma} & := & [\![\neg(\neg\varphi_1 \vee \neg\varphi_2)]\!]^{\mathcal{D}}_{\sigma} \\
[\![\varphi_1 \rightarrow \varphi_2]\!]^{\mathcal{D}}_{\sigma} & := & [\![\neg\varphi_1 \vee \varphi_2]\!]^{\mathcal{D}}_{\sigma} \\
[\![\varphi_1 \leftrightarrow \varphi_2]\!]^{\mathcal{D}}_{\sigma} & := & [\![(\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)]\!]^{\mathcal{D}}_{\sigma} \\
[\![\forall x\varphi]\!]^{\mathcal{D}}_{\sigma} & := & [\![\neg\exists x\,\neg\varphi]\!]^{\mathcal{D}}_{\sigma}
\end{array}
$$

FOL Semantic: kingship example

- Possible extended interpretation:
  - Intended interpretation (see before) +
    - ∀x King(x) → Person(x)
    - $\exists x \;\; Crown(x) \wedge OnHead(x, John)$
    - **N.B.**
      - → (and not ∧) is the natural connective to use with ∀
        - Using ∧ as the main connective with ∀ lead to an overly strong statement
      - ∧ (and not →) is the natural connective to use with ∃
        - using → with ∃ usually leads to a very weak statement

        - Because '→' is True whenever its premise is False

Recall from exercise 1: Aristotelian forms

- All Ps are Qs (where P, Q are two unary relations)
  - $\forall x.\ (P(x) \rightarrow Q(x))$
- No Ps are Qs
  - $\forall x.\ (P(x) \rightarrow \neg Q(x))$
- Some Ps are Qs
  - $\exists x.\ (P(x) \wedge Q(x))$
  - Remark: same sentence as “some Qs are Ps”
- Some Ps aren’t Q’s
  - $\exists x.\ (P(x) \wedge \neg Q(x))$

FOL Semantic

- The extended interpretation map quantifier variables to objects in the model, define the truth of quantified sentences

- **Satisfiability**
  - A sentence α is satisfiable if it is true under some interpretation (D, σ)
- **Model (*a.k.a.* world)**
  - An interpretation (D,σ) is a **model** of a sentence α if α is true under (D,σ)
  - Then we write (D,σ) |= α
- A sentence α is **valid** in D if (D,σ) |= α for all σ
- A sentence is **unsatisfiable** if it has no model

Exercise

- Consider the universe N of natural numbers
  - $\exists x.x+1>5$ is satisfiable or unsatisfiable in N?
    - Solution:
  - $\forall x.x+1>0$ is valid or non-valid in N?
    - Solution:
  - $\exists x.2x+1=6$ is satisfiable or unsatisfiable in N?
    - Solution:

Summary

- The syntax of first-order logic builds on that of propositional logic, with the addition of objects, quantifiers, and =
- A possible world (or model) includes a set of objects and an interpretation that maps constant symbols to objects, predicate symbols to relations among objects, and function symbols to functions on objects
- An atomic sentence is true just when the relation named by the predicate holds between the objects named by the terms
- Extended interpretations, which map quantifier variables to objects in the model, define the truth of quantified sentences
- Developing a KB in first-order logic requires a careful process of analyzing the domain, choosing a vocabulary, and encoding the axioms required to support the desired inferences

First-order logic

Inference in first-order logic: resolution

Recall: inference in propositional logic

- In propositional logic, inference is based on the modus ponens:
  - *From p and p -> q, one can deduce q*

$$\frac{p \qquad p \rightarrow q}{q} \equiv \frac{p \qquad (\neg p \vee q)}{q}$$

- Resolution techniques (including forward chaining) are **complete** for KBs in the Horn form, *e.g.* $(A \vee \neg B) \wedge (\neg A \vee \neg C \vee D)$

- Resolution in propositional logic

$$\frac{(p \vee L_1 \vee \ldots L_n) \qquad (\neg p \vee M_1 \vee \cdots \vee M_k)}{(L_1 \vee \ldots L_n \vee M_1 \vee \cdots \vee M_k)}$$

Resolvent of $(p \vee L_1 \vee \ldots L_n)$ and $(\neg p \vee M_1 \vee \cdots \vee M_k)$

  - Permuting the literals does not change anything

Inference in FOL

- In FOL, the same resolution techniques can be used, but there are some difficulties
  - Quantifiers
  - Infinite sets of terms
  - Infinite sets of sentences
- Examples: $\forall x.\ (King(x) \land Greedy(x) \rightarrow Evil(x))$
  - Infinite set of instances
    - $King(Bill) \land Greedy(Bill) \rightarrow Evil(Bill)$
    - $King(FatherOf(Bill)) \land Greedy(FatherOf(Bill)) \rightarrow Evil(FatherOf(Bill))$

Inference in FOL

- So, in FOL, the KB is expressed using **predicates**
  - Recall: If $t_1,\ldots, t_n$ are terms (composed of objects, constants, and functions) and P is a relation symbol with arity n, then $P(t_1,\ldots, t_n)$ is a predicate
    - Tall(FatherOf(Bill)), Odd(X), Married(Tom,Marry), Loves(Y,MotherOf(Y)), …

- We can use the same resolution as in propositional language, but in FOL we don't look for **equal** (positive / negative) clauses in the KB
  - Instead, we look for **unifiable** clauses
  - For instance:

$$\frac{P(a) \qquad (\neg P(x) \vee Q(x))}{Q(a)}$$

  - Because P(a) and P(x) are **unifiable** under the **substitution** $x \mapsto a$
    - Notion of **Most General Unifier (MGU)**

Inference in FOL

- More generally, resolution in FOL can be expressed as:

$$\frac{(A \vee L_1 \vee \dots L_n) \qquad (\neg B \vee M_1 \vee \dots \vee M_k)}{(L_1[\sigma] \vee \dots L_n[\sigma] \vee M_1[\sigma] \vee \dots \vee M_k[\sigma])}$$

  - Where **σ** is a **principal unifier** of the predicates A and B (supposing that they are unifiable)
    - Intuitively, an interpretation where A and B are the same
  - **Unification**: examples:
    - P(x,A,y) and P(C,A,z) where A,C are constants, are unifiable under the **substitution** x ↦ C, y ↦ z
    - But, we can substitute only variables (constants cannot be substituted), so
      - P(x,A,y) and P(C,B,z) are not unifiable if A, B and C are constants

Inference in FOL

- More generally, resolution in FOL can also be expressed as:

$$\frac{A \vee B \qquad \neg C \vee D}{\theta(A \vee D)} \qquad \theta = mgu(B, C)$$

  - mgu: most general unifier
    - The most general assignment of variables to terms in such a way that two terms are equal
    - Syntactical unification algorithm (see Appendix)

Example of Resolution rule

- x, y are variables
- a, b are constants (we should write A, B, but some authors also use small letters for constants)

$$\frac{P(x) \vee Q(x,a) \qquad \neg Q(b,y) \vee R(y)}{P(b) \vee R(a)} \qquad \theta = \{x \mapsto b, y \mapsto a\}$$

Exercise 1

- What is the resolvent of $C_1 = \neg P(x) \vee \neg Q(y) \vee R(x,y)$?

  $C_2 = Q(a)$

  $C_3 = P(b)$

  - Solution:

First-order logic

Inference in first-order logic:

Robinson's resolution

Robinson's Resolution

- **Herbrand's** Theorem (~1930)
  - "A set of sentences S is unsatisfiable if and only there exists a finite subset S_g of the set of all ground instances Gr(S), which is unsatisfiable"
  - Herbrand thus showed that there is a procedure to demonstrate the unsatisfiability of an unsatisfiable set of sentences
- **Robinson** propose the Resolution procedure (~1950)
  - Sound and complete procedure for checking the unsatisfiability of a set of clauses
  - often used for **model checking** applications
    - For instance, model checking is widely used for the verification of hardware and software in industry

Idea of Robinson's Resolution

- Refutation-based procedure
  - S: a set of sentences and A a sentence (theorem)
  - *S |= A:* (recall about logical entailment with this small tutorial : https://www.youtube.com/watch?v=2M-K5OjVgYQ)
  - *S |= A* if and only if S ∩ ¬A is unsatisfiable, *i.e.*

$$S \cap \neg A \models \text{False}$$

- Resolution procedure (by refutation)
  - Transform S ∩ ¬A into a CNF set of clauses
    - Recall: a sentence is in conjunctive normal form (CNF) if it is a **conjunction** of one or more clauses, where a clause is a **disjunction** of literals: it is an AND of ORs
  - Apply Resolution rule to find the empty clause (contradiction)
    - If the empty clause is found
      - Conclude *S |= A*
    - Otherwise
      - No conclusion

Idea of Robinson's Resolution

- Key facts: it is possible to convert any KB into CNF
  - Example:
    - $A \wedge B \rightarrow C$
    - Equivalent to: $\neg (A \wedge B) \vee C$
    - Equivalent to: $(\neg A \vee \neg B) \vee C$
    - Equivalent to: $(\neg A \vee \neg B \vee C)$

Transform a KB into a CNF

1. Eliminate implication and equivalence
2. Move negation inward
3. Standardize variable scope
4. Move quantifiers outward
5. Skolemize existential quantifiers
6. Eliminate universal quantifiers
7. Distribute $\wedge$, $\vee$
8. Flatten $\wedge$, $\vee$
9. Eliminate $\wedge$

1.Eliminate implication

- One can eliminate implication and equivalence by using the following substitutions

$$\begin{array}{lcl} \alpha \rightarrow \beta & \mapsto & \neg\alpha \vee \beta \\ \alpha \leftrightarrow \beta & \mapsto & (\neg\alpha \vee \beta) \wedge (\neg\beta \vee \alpha) \end{array}$$

- Example:

$$\{\forall x\ (\forall y\ P(x,y)) \rightarrow \neg(\forall y\ Q(x,y) \rightarrow R(x,y))\}$$

  - Equivalent to:

$$\{\forall x\ \neg(\forall y\ P(x,y)) \vee \neg(\forall y\ \neg Q(x,y) \vee R(x,y)))\}$$

2. Move negation inward

- One can move negation inward (inside the sentence) by using the following substitutions

$\neg\neg\alpha$	$\mapsto$	α	$\neg\forall v\ \alpha$	$\mapsto$	$\exists v\ \neg\alpha$
$\neg(\alpha \vee \beta)$	$\mapsto$	$\neg\alpha \wedge \neg\beta$	$\neg\exists v\ \alpha$	$\mapsto$	$\forall v\ \neg\alpha$
$\neg(\alpha \wedge \beta)$	$\mapsto$	$\neg\alpha \vee \neg\beta$			

- Example:

$$\{\forall x\ \neg(\forall y\ P(x,y)) \vee \neg(\forall y\ \neg Q(x,y) \vee R(x,y)))\}$$

  - Equivalent to:

$$\{\forall x\ (\exists y\ \neg P(x,y)) \vee (\exists y\ Q(x,y) \wedge \neg R(x,y))\}$$

3. Standardize variable scope

- In short, to standardize variable scope is to:

> Use one different variable for each quantifier

- Example:

$$\{\forall x\,(\exists y\,\neg P(x,y)) \vee (\exists y\,Q(x,y) \wedge \neg R(x,y))\}$$

  - Equivalent to:

$$\{\forall x\,(\exists y\,\neg P(x,y)) \vee (\exists z\,Q(x,z) \wedge \neg R(x,z))\}$$

4. Move quantifiers outward

- One can move quantifiers outward (at the beginning of the sentence) by using the following substitutions (where Q is the quantifier)

$$
\begin{array}{llll}
(Qx\ \alpha) \wedge \beta & \mapsto & Qx\ (\alpha \wedge \beta) & \alpha \wedge (Qx\ \beta) \mapsto Qx\ (\alpha \wedge \beta) \\
(Qx\ \alpha) \vee \beta & \mapsto & Qx\ (\alpha \vee \beta) & \alpha \vee (Qx\ \beta) \mapsto Qx\ (\alpha \vee \beta)
\end{array}
$$

- Example:

$$\{\forall x\ (\exists y\ \neg P(x,y)) \vee (\exists z\ Q(x,z) \wedge \neg R(x,z))\}$$

  - Equivalent to:

$$\{\forall x\ \exists y\ \exists z\ \neg P(x,y) \vee (Q(x,z) \wedge \neg R(x,z))\}$$

5. Skolemize existential quantifiers

- One can skolemize existential quantifiers by introducing new function symbols as follows:

> For each existentially quantified variable, introduce a n-place function, where n is the number of previously appearing universal quantifiers

- Example:

$$\{\forall x\ \exists y\ \exists z\ \neg P(x,y) \vee (Q(x,z) \wedge \neg R(x,z))\}$$

  - Equivalent to:

$$\{\forall x\ \neg P(x, F_1(x)) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x)))\}$$

6. Eliminate universal quantifiers

- One can eliminate universial quantifiers by using the following substitution

$\forall x\ \alpha \mapsto \alpha$

- Example:

$$\{\forall x\ \neg P(x, F_1(x)) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x)))\}$$

  - Equivalent to:

$$\{\neg P(x, F_1(x)) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x)))\}$$

7. Distribute $\wedge$, $\vee$

- One can distribute $\wedge$ and $\vee$ "in a CNF way" by using the following substitutions

$$\begin{aligned} \alpha \vee (\beta \wedge \gamma) &\mapsto (\alpha \vee \beta) \wedge (\alpha \vee \gamma) \\ (\beta \wedge \gamma) \vee \alpha &\mapsto (\beta \vee \alpha) \wedge (\gamma \vee \alpha) \end{aligned}$$

- Example:

$$\{\neg P(x, F_1(x)) \vee (Q(x, F_2(x)) \wedge \neg R(x, F_2(x)))\}$$

  - Equivalent to:

$$\{(\neg P(x, F_1(x)) \vee Q(x, F_2(x))) \wedge (\neg P(x, F_1(x)) \vee \neg R(x, F_2(x)))\}$$

8. Flatten $\wedge$, $\vee$

- One can flatten $\wedge$ and $\vee$ (remove useless parenthesis) by using the following substitutions

$$
\begin{array}{lcl}
(\alpha \wedge (\beta \wedge \gamma)) & \mapsto & (\alpha \wedge \beta \wedge \gamma) \\
(\alpha \vee (\beta \vee \gamma)) & \mapsto & (\alpha \vee \beta \vee \gamma) \\
((\alpha \wedge \beta) \wedge \gamma) & \mapsto & (\alpha \wedge \beta \wedge \gamma) \\
((\alpha \vee \beta) \vee \gamma) & \mapsto & (\alpha \vee \beta \vee \gamma)
\end{array}
$$

- Example:

$$\{(\neg P(x, F_1(x)) \vee Q(x, F_2(x))) \wedge (\neg P(x, F_1(x)) \vee \neg R(x, F_2(x)))\}$$

  - Is already “flattened”; output of “flattening”:

$$\{(\neg P(x, F_1(x)) \vee Q(x, F_2(x))) \wedge (\neg P(x, F_1(x)) \vee \neg R(x, F_2(x)))\}$$

9. Eliminate $\wedge$

- One can eliminate $\wedge$ by using the following substitution

$$\{\alpha \wedge \beta\} \quad \mapsto \quad \{\alpha,\ \beta\}$$

- Example:

$$\{(\neg P(x, F_1(x)) \vee Q(x, F_2(x))) \wedge (\neg P(x, F_1(x)) \vee \neg R(x, F_2(x)))\}$$

  - Equivalent to

$$\{\neg P(x, F_1(x)) \vee Q(x, F_2(x)),\ \neg P(x, F_1(x)) \vee \neg R(x, F_2(x))\}$$

Transform a KB into a CNF

Important remark: in order to transform a KB into a CNF, all the following steps must be followed **IN THE ORDER HEREAFTER**

1. Eliminate implication and equivalence
2. Move negation inward
3. Standardize variable scope
4. Move quantifiers outward
5. Skolemize existential quantifiers
6. Eliminate universal quantifiers
7. Distribute $\wedge$, $\vee$
8. Flatten $\wedge$, $\vee$
9. Eliminate $\wedge$

Example of proof by Robisnon's resolution (refutation)

- Example:
  - John likes all kind of food.
  - Apple and vegetable are food
  - Anything anyone who is not killed eats is food
  - Anil eats peanuts and still alive
  - Harry eats everything that Anil eats.
- Prove by resolution that:
  - John likes peanuts.

Example of proof by Robisnon's resolution (refutation)

- Step 1: conversion to FOL:
  a. John likes all kind of food
  b. Apple and vegetable are food
  c. Anything anyone who is not killed eats is food
  d. Anil eats peanuts and Anil is still alive
  e. Harry eats everything that Anil eats

a. $\forall x$: food(x) → likes(John, x)
b. food(Apple) ∧ food(vegetables)
c. $\forall x \forall y$: eats(x, y) ∧ ¬ killed(x) → food(y)
d. eats (Anil, Peanuts) ∧ alive(Anil).
e. $\forall x$: eats(Anil, x) → eats(Harry, x)

- Prove by resolution that:
  h. John likes peanuts

h. likes(John, Peanuts)

Example of proof by Robisnon's resolution (refutation)

- Step 2: Addition of new predicates in the KB

a. John likes all kind of food

b. Apple and vegetable are food

c. Anything anyone who is not killed eats is food

d. Anil eats peanuts and Anil is still alive

e. Harry eats everything that Anil eats

a. $\forall x$: food(x) $\rightarrow$ likes(John, x)

b. food(Apple) $\wedge$ food(vegetables)

c. $\forall x \forall y$: eats(x, y) $\wedge \neg$ killed(x) $\rightarrow$ food(y)

d. eats (Anil, Peanuts) $\wedge$ alive(Anil).

e. $\forall x$: eats(Anil, x) $\rightarrow$ eats(Harry, x)

f. $\forall x$: $\neg$ killed(x) $\rightarrow$ alive(x)

g. $\forall x$: alive(x) $\rightarrow \neg$ killed(x)

} **added predicates.**

Example of proof by Robisnon's resolution (refutation)

- Step 3: Conversion of FOL into CNF

a. $\forall x$: food(x) $\rightarrow$ likes(John, x)

b. food(Apple) $\wedge$ food(vegetables)

c. $\forall x \forall y$: eats(x, y) $\wedge \neg$ killed(x) $\rightarrow$ food(y)

d. eats (Anil, Peanuts) $\wedge$ alive(Anil).

e. $\forall x$: eats(Anil, x) $\rightarrow$ eats(Harry, x)

f. $\forall x$: $\neg$ killed(x) $\rightarrow$ alive(x) } **added predicates.**

g. $\forall x$: alive(x) $\rightarrow \neg$ killed(x) }

1. **Eliminate all implication ($\rightarrow$) and rewrite**
   a. $\forall x \neg$ food(x) V likes(John, x)
   b. food(Apple) $\wedge$ food(vegetables)
   c. $\forall x \forall y \neg$ [eats(x, y) $\wedge \neg$ killed(x)] V food(y)
   d. eats (Anil, Peanuts) $\wedge$ alive(Anil)
   e. $\forall x \neg$ eats(Anil, x) V eats(Harry, x)
   f. $\forall x \neg$ [$\neg$ killed(x)] V alive(x)
   g. $\forall x \neg$ alive(x) V $\neg$ killed(x)

Example of proof by Robisnon's resolution (refutation)

- Step 3: Conversion of FOL into CNF

a. $\forall x \neg food(x) \vee likes(John, x)$

b. $food(Apple) \wedge food(vegetables)$

c. $\forall x \forall y \neg [eats(x, y) \wedge \neg killed(x)] \vee food(y)$

d. $eats(Anil, Peanuts) \wedge alive(Anil)$

e. $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

f. $\forall x \neg [\neg killed(x)] \vee alive(x)$

g. $\forall x \neg alive(x) \vee \neg killed(x)$

2. **Move negation (¬)inwards and rewrite**

a. $\forall x \neg food(x) \vee likes(John, x)$

b. $food(Apple) \wedge food(vegetables)$

c. $\forall x \forall y \neg eats(x, y) \vee killed(x) \vee food(y)$

d. $eats(Anil, Peanuts) \wedge alive(Anil)$

e. $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

f. $\forall x \neg killed(x)] \vee alive(x)$

g. $\forall x \neg alive(x) \vee \neg killed(x)$

Example of proof by Robisnon's resolution (refutation)

- Step 3: Conversion of FOL into CNF

a. $\forall x \neg food(x) \vee likes(John, x)$

b. $food(Apple) \wedge food(vegetables)$

c. $\forall x \forall y \neg eats(x, y) \vee killed(x) \vee food(y)$

d. $eats(Anil, Peanuts) \wedge alive(Anil)$

e. $\forall x \neg eats(Anil, x) \vee eats(Harry, x)$

f. $\forall x \neg killed(x)] \vee alive(x)$

g. $\forall x \neg alive(x) \vee \neg killed(x)$

3. **Rename variables or standardize variables**

a. $\forall x \neg food(x) \vee likes(John, x)$

b. $food(Apple) \wedge food(vegetables)$

c. $\forall y \forall z \neg eats(y, z) \vee killed(y) \vee food(z)$

d. $eats(Anil, Peanuts) \wedge alive(Anil)$

e. $\forall w \neg eats(Anil, w) \vee eats(Harry, w)$

f. $\forall g \neg killed(g)] \vee alive(g)$

g. $\forall k \neg alive(k) \vee \neg killed(k)$

Example of proof by Robisnon's resolution (refutation)

- Step 3: Conversion of FOL into CNF

a. ∀x ¬ food(x) V likes(John, x)

b. food(Apple) Λ food(vegetables)

c. ∀y ∀z ¬ eats(y, z) V killed(y) V food(z)

d. eats (Anil, Peanuts) Λ alive(Anil)

e. ∀w¬ eats(Anil, w) V eats(Harry, w)

f. ∀g ¬killed(g)] V alive(g)

g. ∀k ¬ alive(k) V ¬ killed(k)

4. Move quantifiers outward: not needed here (already verified)
5. Skolemize existential quantifiers: not needed here (already no existential quantifiers)

Example of proof by Robisnon's resolution (refutation)

- Step 3: Conversion of FOL into CNF

a. $\forall x \neg$ food(x) V likes(John, x)

b. food(Apple) $\wedge$ food(vegetables)

c. $\forall y \forall z \neg$ eats(y, z) V killed(y) V food(z)

d. eats (Anil, Peanuts) $\wedge$ alive(Anil)

e. $\forall w \neg$ eats(Anil, w) V eats(Harry, w)

f. $\forall g \neg$killed(g)] V alive(g)

g. $\forall k \neg$ alive(k) V $\neg$ killed(k)

6. Eliminate universal quantifiers (since all the statements are not implicitly quantified), then

7., 8. Distribute then flatten $\wedge$, $\vee$
(will not change anything)

9. Eliminate $\wedge$ (decompose KB)

a. $\neg$ food(x) V likes(John, x)

b. food(Apple)

c. food(vegetables)

d. $\neg$ eats(y, z) V killed(y) V food(z)

e. eats (Anil, Peanuts)

f. alive(Anil)

g. $\neg$ eats(Anil, w) V eats(Harry, w)

h. killed(g) V alive(g)

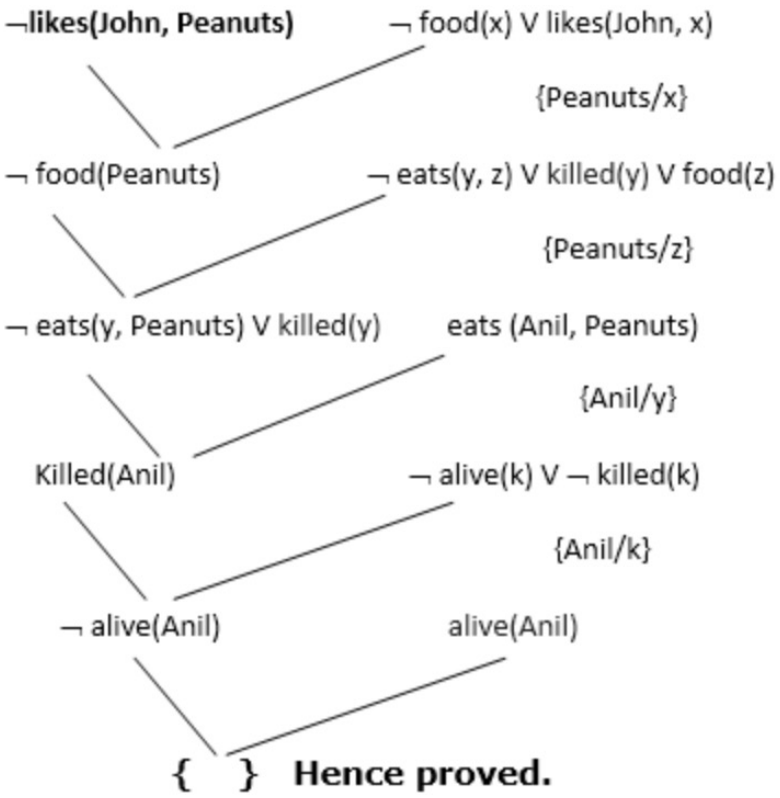
i. $\neg$ alive(k) V $\neg$ killed(k)

Example of proof by Robisnon's resolution (refutation)

- Step 4: Draw resolution graph for proving by refutation that likes(John, Peanuts).

Given the following KB:

a. ¬ food(x) V likes(John, x)
b. food(Apple)
c. food(vegetables)
d. ¬ eats(y, z) V killed(y) V food(z)
e. eats (Anil, Peanuts)
f. alive(Anil)
g. ¬ eats(Anil, w) V eats(Harry, w)
h. killed(g) V alive(g)
i. ¬ alive(k) V ¬ killed(k)



First-order logic

Properties

Properties of First-Order Logic

- Using the resolution procedure above, FOL is:
  - **Complete**: Every necessarily true statement is provable, and every necessarily false statement has a proof of its negation
  - **Undecidable**: there are statements which are neither necessarily true, nor necessarily false.
    - For example, $\forall x\ P(x)$ may be either true or false depending on the universe of discourse and the definition of P
      - On such a statement, the resolution algorithm would never halt, because it would never find a proof or a disproof; but at no point could we be sure that no proof or disproof exists.

First-order logic

Homework

Exercice 1

- Jack owns a dog
- Every dog owner is an animal lover
- No animal lover kills an animal
- Either Jack or Curiosity killed the cat, who is named Tuna
- Did Curiosity kill the cat?

Exercice 2

- The law says that it is a crime for an American to sell weapons to hostile nations
- The country Nono, an enemy of America, has some missiles, and all of its missiles were sold to it by Colonel West, who is American
- Is West a criminal?

First-order logic

Appendix: unification

Appendix: Unification

- Input
  - Set of equalities between two terms
- Output
  - Most general assignment of variables that satisfies all equalities
  - Fail if no such assignment exists

Appendix: Unification algorithm

Decompose

$$U \cup \{f(t_1, \ldots, t_n) =^? f(s_1, \ldots, s_n)\} \longrightarrow U \cup \{t_1 =^? s_1, \ldots, t_n =^? s_n\}$$

Orient.

$$U \cup \{t =^? v\} \longrightarrow U \cup \{v =^? t\}$$

Delete.

$$U \cup \{v =^? v\} \longrightarrow U$$

Eliminate.

$$U \cup \{v =^? t\},\ v \in \mathcal{V}ars(U) \setminus \mathcal{V}ars(t) \longrightarrow U[v/t] \cup \{v =^? t\}$$

Mismatch.

$$U \cup \{f(t_1, \ldots, t_m) =^? g(s_1, \ldots, s_n)\},\ f, g \text{ distinct or } m \neq n \longrightarrow FAIL$$

Occurs.

$$U \cup \{v =^? t\},\ v \neq t \text{ but } v \in \mathcal{V}ars(t) \longrightarrow FAIL$$

- Vars(U), Vars(t) are sets of variables in U and t
- v is a variable
- s and t are terms
- f and g are function symbols

Appendix: Example of Unification

$$\{\underline{F(G(H(y)), H(A)) =^? F(G(x), x)}\} \xrightarrow{\text{Decompose}}$$

$$\{\underline{G(H(y)) =^? G(x)},\ H(A) =^? x\} \xrightarrow{\text{Decompose}}$$

$$\{\underline{H(y) =^? x},\ H(A) =^? x\} \xrightarrow{\text{Orient}}$$

$$\{\underline{x =^? H(y)},\ H(A) =^? x\} \xrightarrow{\text{Eliminate } x}$$

$$\{x =^? H(y),\ \underline{H(A) =^? H(y)}\} \xrightarrow{\text{Decompose}}$$

$$\{x =^? H(y),\ \underline{A =^? y}\} \xrightarrow{\text{Orient}}$$

$$\{x =^? H(y),\ \underline{y =^? A}\} \xrightarrow{\text{Eliminate } y}$$

$$\{x =^? H(A),\ y =^? A\}$$

Chapter 4 – part 3

Questions

ĐẠI HỌC BÁCH KHOA

25 YEARS ANNIVERSARY SOICT

VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

SCHOOL OF INFORMATION AND COMMUNICATION TECHNOLOGY

Thank you for your attention!

soict.hust.edu.vn/ fb.com/groups/soict