AIFA
First Order Logic

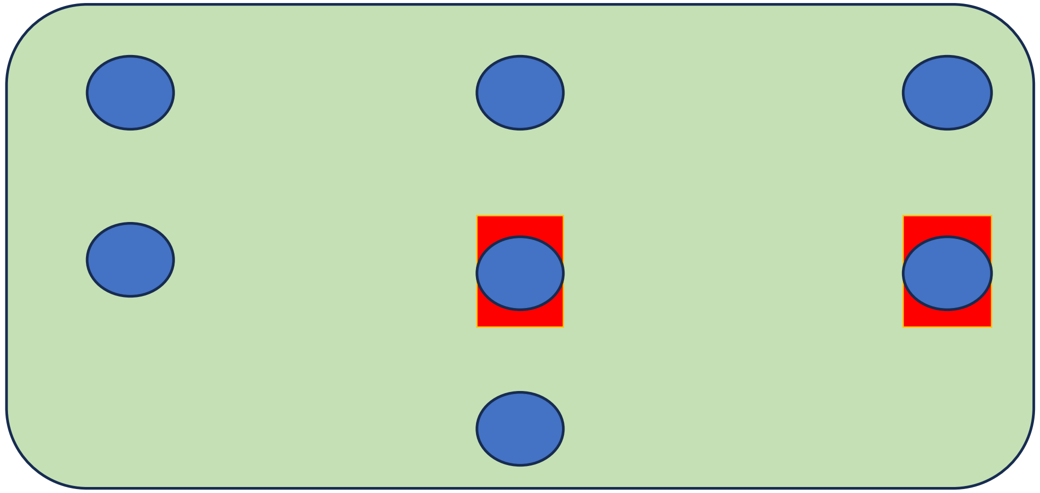
30/01/2024

Koustav Rudra

Quantifiers

- There are two basic quantifiers in FOL

- $\forall$ "for all" Universal Quantifier
- $\exists$ "there exists" Existential Quantifier



Days in a week

$$\forall_x sunrise(x)$$

$$\exists_x holiday(x)$$

Universal Quantifiers

- All dogs are faithful
  - faithful(x): x is faithful
  - dog(x): x is a dog
  - $\forall_x(\text{dog}(x) \rightarrow \text{faithful}(x))$

- All birds cannot fly
  - fly(x): x can fly
  - bird(x): x is a bird
  - $\boxed{\sim}(\forall_x(\text{bird}(x) \rightarrow \text{fly}(x)))$

Existential Quantifiers

- At least one planet has life on it
  - planet(x): x is a planet
  - haslife(x): x has life on it
  - $\exists_x$(planet(x)∧haslife(x))

- There exists a bird that can't fly
  - fly(x): x can fly
  - bird(x): x is a bird
  - $\exists_x$(bird(x)∧~fly(x))

First Order Logic

30/01/2024

Koustav Rudra

Duality of Quantifiers

- All men are mortal
  - No man is immortal

- $\forall_x(\text{man}(x) \rightarrow \text{mortal}(x))$
- $\boxed{\sim}(\exists_x(\text{man}(x) \wedge \sim \text{mortal}(x)))$

Universal quantifiers could be expressed in a different way with existential quantifiers

Sentences

- A predicate is a sentence

- If sen, sen' are sentences and x is a variable then following are sentences
  - (sen), ~sen, $\exists_x$sen, $\forall_x$sen
  - sen$\wedge sen'$
  - sen$\vee sen'$
  - sen$\rightarrow sen'$

First-order Logic

- Sentence →AtomicSentence
  - | Sentence Connective Sentence
  - Quantifier Variable, … Sentence
  - ~ Sentence | (Sentence)

- AtomicSentence → Predicate(Term, …)
  - | Term = Term
- Term →Function(Term, …) | Constant | Variable
- Connective → ⇒|∨|∧|⇔
- Quantifier →∀|∃

Difference from second order logic

- In FOL we quantify only variables
- In second order logic we can quantify predicates
  - $\exists_P \forall_x \forall_y P(x, y)$

Examples

- Not all students take history and biology
  - Student(x): x is a student
  - Takes(x,y): subject x is taken by y
  - $\boxed{\sim}$[$\forall_x$Student(x)→Takes(History,x) ∧ Takes(Biology,x)]
  - $\exists_x$Student(x) ∧ [~Takes(History,x) ∨ ~Takes(Biology,x)]

- Only one student failed biology
  - Failed(x,y): student y failed in subject x
  - $\exists_x$[Student(x) ∧ Failed(biology,x) ∧
  - $\forall_y$[(~(x=y) ∧ Student(y))→~Failed(biology,y)]]

Examples

- Only one student failed both history and biology
  - Failed(x,y): student y failed in subject x
  - $\exists_x$[Student(x) $\wedge$ Failed(biology,x) $\wedge$ Failed(history,x) $\wedge$
  - $\forall_y$[(~(x=y) $\wedge$ Student(y))$\rightarrow$~Failed(biology,y) $\vee$ ~Failed(history,y)]]

- The best score in history is better than the best score in biology
  - Function: score(subject, student)
  - Greater(x,y): x>y
  - $\forall_x$[Student(x) $\wedge$ Takes(biology,x) $\rightarrow$ $\exists_y$ [Student(y) $\wedge$ Takes(history,y) $\wedge$ Greater(score(history,y),score(biology,x))]]
  - $\exists_x$[Student(x) $\wedge$ Takes(biology,x) $\wedge$ $\exists_y$ [Student(y) $\wedge$ Takes(history,y) $\wedge$ Greater(score(history,y),score(biology,x))]]

Example of sentences

- Brother(x,y): x is y's brother
- Loves(x,y): x loves y

- $\forall_x \forall_y$Brother(x,y)→Loves(x,y)
  - Everyone loves his/her brothers

- A sentence is evaluated to be true in a particular domain, the sentence is said to be satisfied
- A sentence is valid in terms of FOL, for all possible domains this sentence will evaluate to be true

Takeaway

- Predicate logic is a more powerful version of proposition logic
- Provide support for variables and quantifiers

- Can capture sentences more naturally

How we can perform inferencing with predicates?

Inference in FOPL

30/01/2024

Koustav Rudra

Inference Rules

- Universal Elimination
    - $\forall_x$Likes(x, flower)
    - Substituting x by Akash {x/Akash}
        - Likes(Akash, flower)

- The substitution should be done by a <u>constant term</u>

Inference Rules

- Existential elimination (Skolemization)
  - $\exists_x$Likes(x, flower) ➔Likes(Person, flower)
    - as long as Person is not in the knowledge base

- This method of finding out a particular constant that satisfies the predicate there exists x that likes flower is known as Skolemization.

- Existential introduction
  - Likes(John, flower)
  - Can be written as
    - $\exists_x$Likes(x, flower)

Example1: Reasoning in FOL

- The law says that it is a crime for a Gaul to sell potion formulas to hostile nations.

- The country Rome, an enemy of Gaul, has acquired some potion formulas, and all of its formulas were sold to it by Druid Traitorix.

- Traitorix is a Gaul

- Is Traitorix a criminal?

Example1: Reasoning in FOL

- The law says that it is a crime for a Gaul to sell potion formulas to hostile nations.
- Gaul(x): x is a Gaul
- Hostile(z): z is hostile nation
- Potion(y): y is a potion
- Criminal(x): x is criminal
- Sells(x,y,z): x sells y to z
- Owns(x,y): x owns y

- $\forall_x \forall_y \forall_z$ Gaul(x) $\wedge$ Potion(y) $\wedge$ Sells(x,y,z) $\wedge$ hostile(z) $\rightarrow$ Criminal(x)

Example1: Reasoning in FOL

- The country Rome, an enemy of Gaul, has acquired some potion formulas, and all of its formulas were sold to it by Druid Traitorix.
- Gaul(x): x is a Gaul
- Hostile(z): z is hostile nation
- Potion(y): y is a potion
- Criminal(x): x is criminal
- Sells(x,y,z): x sells y to z
- Owns(x,y): x owns y

- Hostile(Rome)
- $\exists_y$ Potion(y) ∧ Owns(Rome,y)
- $\forall_y$ Potion(y) ∧ Owns(Rome,y) → Sells(Traitorix, y, Rome)

Example1: Reasoning in FOL

- Traitorix is a Gaul

- Gaul(x): x is a Gaul
- Hostile(z): z is hostile nation
- Potion(y): y is a potion
- Criminal(x): x is criminal
- Sells(x,y,z): x sells y to z
- Owns(x,y): x owns y

- Gaul(Traitorix)
- ?- Criminal(Traitorx) [We have to deduce it]

Example1: Reasoning in FOL

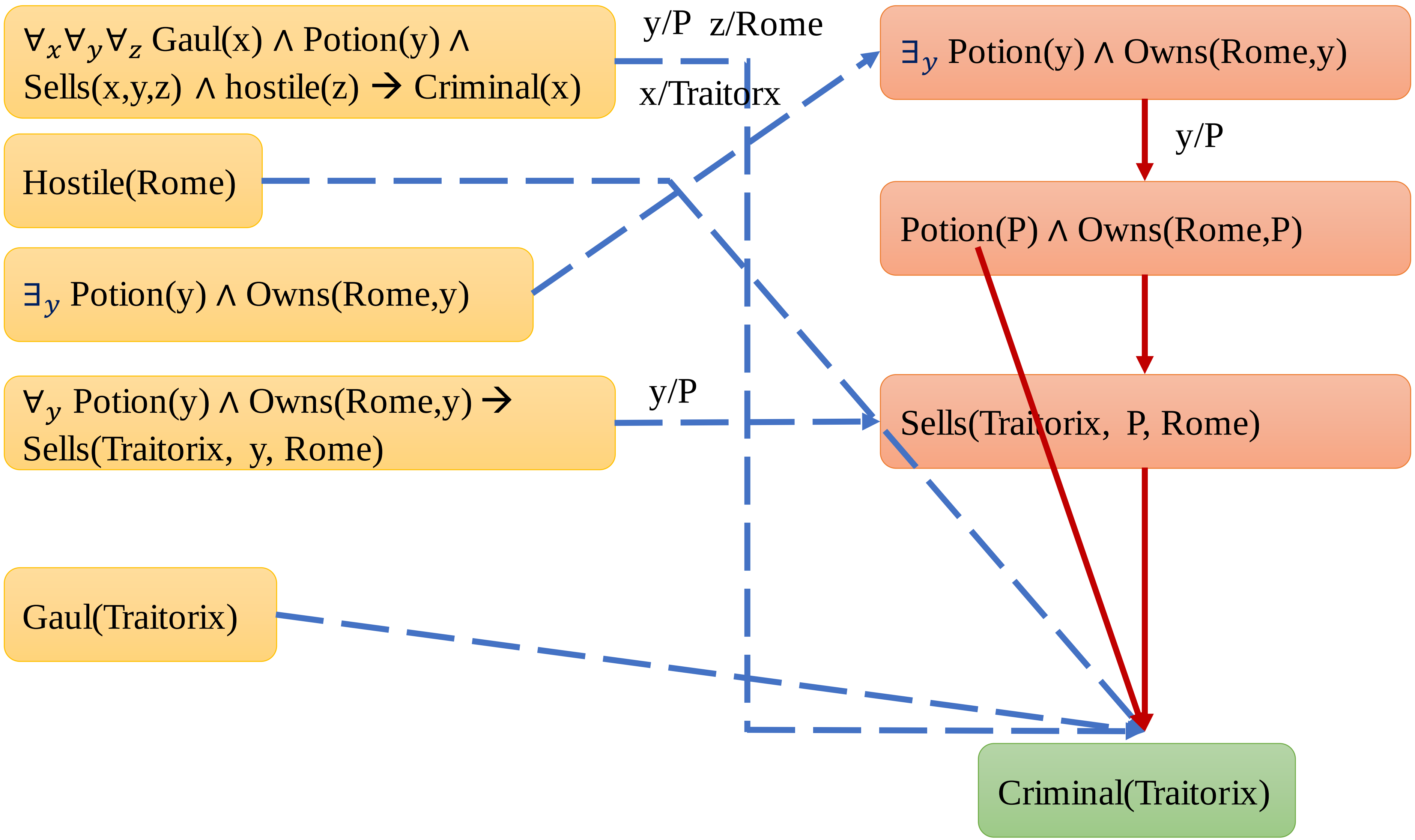
- $\forall_x \forall_y \forall_z$ Gaul(x) ∧ Potion(y) ∧ Sells(x,y,z) ∧ hostile(z) → Criminal(x)
- Hostile(Rome)
- $\exists_y$ Potion(y) ∧ Owns(Rome,y)
- $\forall_y$ Potion(y) ∧ Owns(Rome,y) → Sells(Traitorix, y, Rome)
- Gaul(Traitorix)

- ?- Criminal(Traitorx)

Forward Chaining → Started from existing facts and rules in knowledge base and reach the goal

Backward Chaining → Started from the goal and use existing facts and rules in knowledge base to prove goal

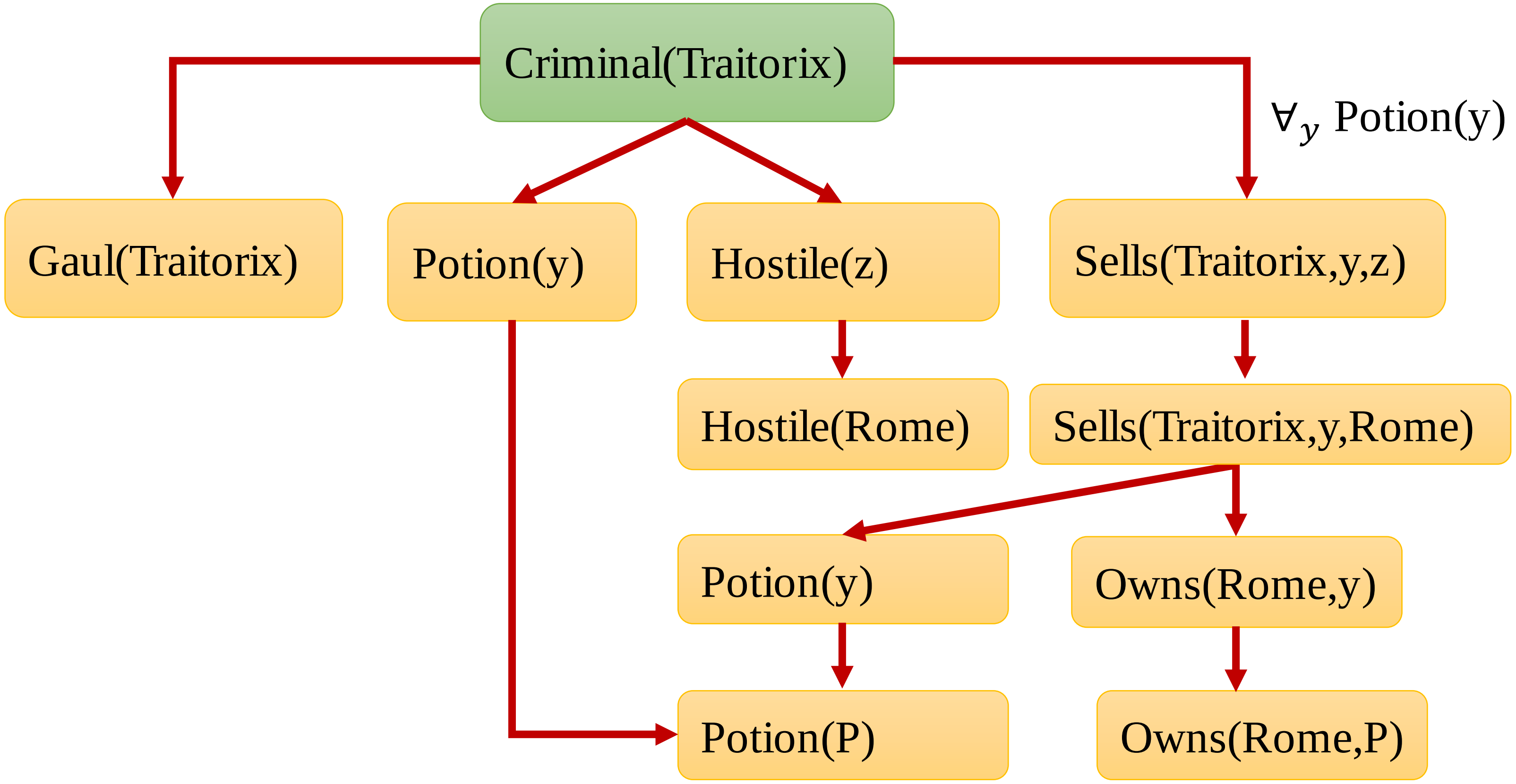
Reasoning in FOL: Forward Chaining



Reasoning in FOL: Backward Chaining

- $\forall_x \forall_y \forall_z$ Gaul(x) ∧ Potion(y) ∧ Sells(x,y,z) ∧ hostile(z) → Criminal(x)
- $\forall_y \forall_z$ Gaul(Traitorix) ∧ Potion(y) ∧ Sells(Traitorix,y,z) ∧ hostile(z) → Criminal(Traitorix)

Hostile(Rome)



$\forall_y$ Potion(y) ∧ Owns(Rome,y) → Sells(Traitorix, y, Rome)

$\exists_y$ Potion(y) ∧ Owns(Rome,y)

Thank You