First-Order Logic (FOL)

Sven Koenig, USC

Russell and Norvig, 3rd Edition, Chapter 8 and Section 9.5

These slides are new and can contain mistakes and typos.
Please report them to Sven (skoenig@usc.edu).

Propositional Logic

- How to represent facts matters:
- P ≡ "2 is prime."
  Q ≡ "2 is even."
  R ≡ "2 is prime and 2 is even."
- P AND Q ⊨ P – yes
- R ⊨ P – no

Propositional Logic

- How to represent facts matters:
- P ≡ "All cats are mammals."
  Q ≡ "Bob is a cat."
  R ≡ "Bob is a mammal."
- "All cats are mammals." AND "Bob is a cat." ⊨ "Bob is a mammal." – yes
  P AND Q ⊨ R – no

First-Order Logic

- In first-order logic, sentences represent propositions.
- Sentences can refer to objects, their relationships and their properties.
- First-order logic is a superset of propositional logic.

- Constants (refer to objects): Adam, Bob, …
- Predicates (refer to relations or properties): IsMotherOf, IsHappy, …
- Functions: MotherOf, …

Syntax

- Constant -> A | … | Z | Adam | Bob | …
- Variable -> a | … | z
- Predicate -> IsMotherOf | IsHappy | …
- Function -> MotherOf | …

Syntax

- Sentence → AtomicSentence | Sentence Connective Sentence | Quantifier Variable Sentence | NOT Sentence | (Sentence)
- AtomicSentence → T(RUE) | F(ALSE) | Predicate (Term, …, Term) | Predicate | Term = Term
- Term → Constant | Variable | Function (Term, …, Term)
- Connective → AND | OR | IMPLIES | EQUIV
- Quantifier → FORALL | EXISTS

Quantifiers

- Two quantifiers
  - FORALL (Universal quantifier, write: "∀", read: "for all")
  - EXISTS (Existential quantifier, write: "∃", read: "there exists")
- Every variable must be introduced by a quantifier before it is used.
- FORALL x IsHappy(x) ≡ IsHappy(Adam) AND IsHappy(Bob) AND …
  "Every object is happy."
- EXISTS x IsHappy(x) ≡ IsHappy(Adam) OR IsHappy(Bob) OR …
  "At least one object is happy."

Quantifiers

- Not every object is happy.
  NOT FORALL x IsHappy(x) ≡
  NOT (IsHappy(Adam) AND IsHappy(Bob) AND …) ≡
  NOT IsHappy(Adam) OR NOT IsHappy(Bob) OR … ≡
  EXISTS x NOT IsHappy(x)
- Every object is happy.
  FORALL x IsHappy(x) ≡
  NOT NOT FORALL x IsHappy(x) ≡
  NOT EXISTS x NOT IsHappy(x)

Quantifiers

- No object is happy.
  FORALL x NOT IsHappy(x) ≡
  NOT IsHappy(Adam) AND NOT IsHappy(Bob) AND … ≡
  NOT(IsHappy(Adam) OR IsHappy(Bob) OR …) ≡
  NOT EXISTS x IsHappy(x)
- At least one object is happy.
  EXISTS x IsHappy(x) ≡
  NOT NOT EXISTS x IsHappy(x) ≡
  NOT FORALL x NOT IsHappy(x)

Quantifiers

- Rewrite rules
  FORALL x P(x) ≡ NOT EXISTS x NOT P(x)
  EXISTS x P(x) ≡ NOT FORALL x NOT P(x)
  FORALL x FORALL y P(x,y) ≡ FORALL y FORALL x P(x,y)
  EXISTS x EXISTS y P(x,y) ≡ EXISTS y EXISTS x P(x,y)
  NOT FORALL x (P(x) IMPLIES Q(x)) ≡ EXISTS x (P(x) AND NOT Q(x))
  NOT EXISTS x (P(x) AND Q(x)) ≡ FORALL x (P(x) IMPLIES NOT Q(x))
  and many more

Quantifiers

- Every person is happy.
  FORALL x (IsPerson(x) IMPLIES IsHappy(x))
- Every object is a happy person
  (= probably not what you mean to say).
  FORALL x (IsPerson(x) AND IsHappy(x))
- At least one person is happy.
  EXISTS x (IsPerson(x) AND IsHappy(x))
- At least one object is not a person or is happy
  (= probably not what you mean to say).
  EXISTS x (IsPerson(x) IMPLIES IsBlue(x))

Quantifiers

- Every object has at least one mother.
  FORALL y EXISTS x IsMotherOf(x,y)
- There exists an object that is the mother of every object
  (= probably not what you mean to say).
  EXISTS x FORALL y IsMotherOf(x,y).
- Every person has at least one mother.
  FORALL y (IsPerson(y) IMPLIES EXISTS x IsMotherOf(x,y))
  It is unnecessary (but not incorrect) to use:
  FORALL y (IsPerson(y) IMPLIES EXISTS x (IsPerson(x) AND IsMotherOf(x,y)))

Examples

- Sue is Bill's mother.
- Sue is Bill's and Tom's mother.
- Bill is sad, and so is Tom.
- Either Bill or Tom is happy but not both.
- Sue has at least one child.
- Sue has no children.
- Sue has at most one child.
- All of Sue's kids are happy.
- At least one of Sue's kids is happy.
- None of Sue's kids are happy.

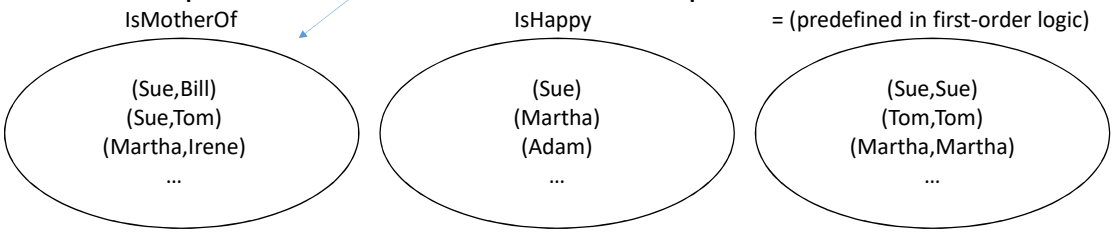
Examples

- IsMotherOf(Sue, Bill)
- IsMotherOf(Sue,Bill) AND IsMotherOf(Sue,Tom)
- IsHappy(Bill) AND IsHappy(Tom).
- IsHappy(Bill) EQUIV NOT IsHappy(Tom)
- EXISTS x IsMotherOf(Sue,x)
- NOT EXISTS x IsMotherOf(Sue,x)
- FORALL x FORALL y (IsMotherOf(Sue,x) AND IsMotherOf(Sue,y) IMPLIES x=y)
- FORALL x (IsMotherOf(Sue,x) IMPLIES IsHappy(x))
- EXISTS x (IsMotherOf(Sue,x) AND IsHappy(x))
- FORALL x (IsMotherOf(Sue,x) IMPLIES NOT IsHappy(x))

Semantics

The set of those tuples that, when used as arguments of IsMotherOf, make it true

- An interpretation (= world = model) assigns each constant an object and each predicate and function a set of tuples.



- Functions are just a convenient way of using predicates where the value of one of the arguments is uniquely determined by the values of the other arguments, e.g.
  FORALL x FORALL y MotherOf(y) = x EQUIV IsMotherOf(x,y)

Semantics

- An interpretation for a sentence (of finite length) in propositional logic can be specified explicitly in finite time and with finite space because the sentence contains a finite number of different symbols. Thus, the computer could be provided with an interpretation although this is not done since it would be inconvenient.
- An interpretation for a sentence (of finite length) in first-order logic cannot necessarily be specified explicitly in finite time and with finite space (for example, the set of those objects that, when used as arguments of IsPrime, make it true is infinite). Thus, the computer cannot be provided with an interpretation.

Semantics

- A sentence (of finite length) in propositional logic contains a finite number n of different symbols and thus has a finite number of interpretations, namely 2^n.
- A sentence (of finite length) in first-order logic can have an infinite number of interpretations (for example, a constant could be assigned the integers 1, 2, 3, …). Thus, one typically cannot enumerate all interpretations, e.g. use truth tables to check whether a sentence is a tautology or whether a knowledge base entails a sentence.

Semantics

- A sentence is …
  - valid (= a tautology)
    if and only if it is true for all interpretations
  - satisfiable
    if and only if it is true for at least one interpretation
  - unsatisfiable (= a contradiction)
    if and only if it is true for no interpretation

Semantics

- Examples:
  IsPrime(2)
  IsPrime(2) OR NOT IsPrime(2)

  KB ≡
  FORALL x Eats(Lion,x) [1]
  AND
  FORALL x FORALL y (Eats(x,y) IMPLIES Hunts(x,y)) [2]
  AND
  FORALL x FORALL y (Hunts(x,y) IMPLIES NOT Hunts(y,x)) [3]

Semantics

- Examples:
  IsPrime(2) – satisfiable
  IsPrime(2) OR NOT IsPrime(2) – valid (and satisfiable)

  KB – unsatisfiable
  [1] entails Eats(Lion,Lion) [4].
  [2] and [4] entail Hunts(Lion,Lion) [5].
  [3] and [5] entail NOT Hunts(Lion,Lion) [6].
  [5] and [6] are a contradiction.

  This was probably a mistake by the knowledge engineer but with bad consequences since an unsatisfiable KB entails everything.

Semantics

- KB ⊨ S (entailment, read: “entails”)
  if and only if, whenever KB is true for an interpretation, then S is also true for that interpretation.

- Examples:
  IsPrime(3) ⊨ IsPrime(2)
  IsPrime(3) ⊨ IsPrime(3)
  IsPrime(3) ⊨ FORALL x IsPrime(x)
  FORALL x IsPrime(x) ⊨ IsPrime(3)

Semantics

- KB ⊨ S (entailment, read: “entails”)
  if and only if, whenever KB is true for an interpretation, then S is also true for that interpretation.

- Examples:
  IsPrime(3) ⊨ IsPrime(2) - no
  IsPrime(3) ⊨ IsPrime(3) - yes
  IsPrime(3) ⊨ FORALL x IsPrime(x) - no
  FORALL x IsPrime(x) ⊨ IsPrime(3) - yes

Resolution

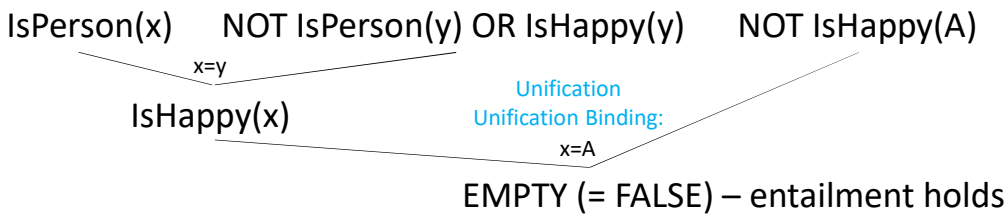
- Using truth tables to prove entailment does not work for first-order logic.
- Using resolution works. As before, we will always use the following scheme:
  - Transform KB AND NOT S into conjunctive normal form.
  - Apply resolution to derive EMPTY (= FALSE).
- Remember that, in propositional logic, we remove duplicates from the clause resulting from a resolution step. Now, we generalize this to removing each literal (predicate or its negation) that can be unified with another literal in the clause and applying the unification binding to the whole clause.
- Please read the textbook on the topic of resolution in first-order logic and go through the examples given there!

Resolution

- When transforming KB AND NOT S into conjunctive normal form:
  - MOVE NOT INWARDS: Move NOT as much inside of quantifiers as possible.
  - Example: Use EXISTS x NOT IsHappy(x) instead of NOT FORALL x IsHappy(x).
  - STANDARDIZE VARIABLES: Use a different variable for each quantifier.
  - Example: Use FORALL x IsHappy(x), FORALL y (IsPerson(y) IMPLIES IsHappy(y)) instead of FORALL x IsHappy(x), FORALL x (IsPerson(x) IMPLIES IsHappy(x)).
  - SKOLEMIZE: Drop each existential quantifier and replace its variable with a new function that uses the variables of the universal quantifiers to its left in its clause as arguments. (Use a constant instead of a function with zero arguments.)
  - Example: Use NOT IsHappy(A) instead of EXISTS x NOT IsHappy(x). Use FORALL y IsMotherOf(M(y),y) instead of FORALL y EXISTS x IsMotherOf(x,y).
  - DROP UNIVERSAL QUANTIFIERS: Drop each universal quantifier.
  - Example: Use IsPerson(x) instead of FORALL x IsPerson(x).

Resolution Example

- FORALL x IsPerson(x) AND FORALL x (IsPerson(x) IMPLIES IsHappy(x)) $\overset{?}{\models}$ FORALL x IsHappy(x)
- FORALL x IsPerson(x), FORALL x (IsPerson(x) IMPLIES IsHappy(x)), EXISTS x NOT IsHappy(x)
- IsPerson(x), IsPerson(y) IMPLIES IsHappy(y), NOT IsHappy(A)



Resolution

- Resolution does not always terminate for first-order logic.
- If KB $\models$ S, then resolution will eventually produce EMPTY (= FALSE).
- If KB $\not\models$ S, then resolution will not produce EMPTY but might not terminate.
- Thus, if resolution has not terminated after x hours (for any x), one does not know whether it will eventually produce EMPTY or not. One can't just wait and see either since it might run forever.
- In general, the question of entailment for first-order logic is only semi-decidable, that is, algorithms exist that say yes to every entailed sentence but no algorithm exists that also says no to every non-entailed sentence. Resolution is an example of such an algorithm.