First-Order Logic

CSE 4308/5360 – Artificial Intelligence I

Vassilis Athitsos
University of Texas at Arlington

Limitations of Propositional Logic

- In the 4x4 wumpus world, how can we say that pits cause breezes in adjacent squares?

Limitations of Propositional Logic

- In the 4x4 wumpus world, how can we say that pits cause breezes in adjacent squares?
  - We need 16 different rules like this:
    B_1_2 <=> (P_1_1 OR P_2_2 OR P_1_3)

Limitations of Propositional Logic

- In the 4x4 wumpus world, how can we say that pits cause breezes in adjacent squares?
  - We need 16 different rules like this:
    B_1_2 <=> (P_1_1 OR P_2_2 OR P_1_3)
- How can we say that adding 1 to an even number produces an odd number?

Limitations of Propositional Logic

- In the 4x4 wumpus world, how can we say that pits cause breezes in adjacent squares?
  - We need 16 different rules like this:
    B_1_2 <=> (P_1_1 OR P_2_2 OR P_1_3)
- How can we say that adding 1 to an even number produces an odd number?
  - We need infinite symbols and infinite rules.
  - A symbol O1 for “1 is odd”, a symbol E2 for “2 is even”, …

Limitations of Propositional Logic

- In the 4x4 wumpus world, how can we say that pits cause breezes in adjacent squares?
  - We need 16 different rules like this:
    B_1_2 <=> (P_1_1 OR P_2_2 OR P_1_3)
- How can we say that adding 1 to an even number produces an odd number?
  - We need infinite symbols and infinite rules.
  - A symbol O1 for “1 is odd”, a symbol E2 for “2 is even”, …
- What do these limitations buy us?
  - Simple syntax: just symbols and connectives.
  - Inference algorithms (like TT-Entails) that are horribly slow (exponential time), but at least terminate in finite time.

First-Order Logic

- In first-order logic, we have a richer language, that can explicitly represent:
  - Objects (called **constants**).
    - John, Mary, house backpack, Arlington, Texas…
  - Relations (also called **predicates**). These are boolean functions (they can only evaluate to true or false).
    - Siblings(John, Mary)
    - >(100, 5)
    - Red(laptop551)
    - Team(John, Mary, Sue, Jim)
  - Functions.
    - Capital(Texas)
    - Mother(John)
    - 25 + 12 (here, + is a function).

Relations and Functions

- Should "sibling" be a relation or a function?
  - Relation: siblings(John, Mary)
  - Function: sibling(John) returns Mary.

Relations and Functions

- Should "sibling" be a relation or a function?
  - Relation: siblings(John, Mary)
  - Function: sibling(John) returns Mary.
- "Sibling" should be a relation, because someone can have many (or no) siblings. A function can only return one value.

Relations and Functions

- Should "sibling" be a relation or a function?
  - Relation: siblings(John, Mary)
  - Function: sibling(John) returns Mary.
- "Sibling" should be a relation, because someone can have many (or no) siblings. A function can only return one value.
- Should "mother" be a relation or a function?
  - Relation: mother(Liz, John)
  - Function: mother(John) returns Liz.

Relations and Functions

- Should "sibling" be a relation or a function?
  - Relation: siblings(John, Mary)
  - Function: sibling(John) returns Mary.
- "Sibling" should be a relation, because someone can have many (or no) siblings. A function can only return one value.
- Should "mother" be a relation or a function?
  - Relation: mother(Liz, John)
  - Function: mother(John) returns Liz.
- "Mother" can be either a relation or a function.
  - A person (or animal) has only one mother.

Basic Elements of First Order Logic

- In propositional logic we only had symbols and connectives.
- In first-order logic we have NO SYMBOLS. Instead, we have:
  - Constants.
  - Predicates.
  - Functions.
  - Connectives (and, or, not, if, iff).
  - The equal sign = (a "special" predicate).
  - Variables.
  - Quantifiers, ∀ (for every), ∃ (there exists).

Variables and Quantifiers

- Variables can only be used together with quantifiers.
- Quantifiers need variables in order to be used.
- Examples:
  - $\forall$ x, y: brothers(x, y) => siblings(x, y)
  - $\exists$ x: 2*5 + x = 18.

Examples

- For the wumpus world, to say that "pits cause breezes in adjacent squares" using propositional logic, we need 16 rules like this:

B_1_2 <=> (P_1_1 OR P_2_2 OR P_1_3)

- In first-order logic, how can we say the same thing?

Examples

- For the wumpus world, to say that "pits cause breezes in adjacent squares" using propositional logic, we need 16 rules like this:

B_1_2 <=> (P_1_1 OR P_2_2 OR P_1_3)

- In first-order logic, how can we say the same thing?

∀ x1, y1: Breeze(x1, y1) <=>
    ∃ x2, y2: Pit(x2, y2) AND Adjacent(x1, y1, x2, y2)

Examples

- For the wumpus world, to say that "there is only one monster" using propositional logic, we need 16 rules like this:

M_2_3 => not(M_1_1 OR M_1_2 OR M_1_3 …)

- In first-order logic, how can we say the same thing?

Examples

- For the wumpus world, to say that "there is only one monster" using propositional logic, we need 16 rules like this:

M_2_3 => not(M_1_1 OR M_1_2 OR M_1_3 …)

- In first-order logic, how can we say the same thing?

∀ x1, y1 : Monster(x1, y1) =>
∀ x2, y2 : Monster(x2, y2) => (x1, y1) = (x2, y2)

Representing Integers

- How do we represent integers in propositional logic?
  - We cannot, at least not explicitly. Propositional logic has no room for objects or constants, just for symbols.
  - We can only represent **properties** of integers. We can use symbols to represent boolean statements about integers.
  - Example: symbol O143 can represent the statement “143 is an odd number”, which is a true statement.
  - Example: symbol P143 can represent the statement “143 is a prime number”, which is a false statement.
- Overall, we need an infinite number of symbols and rules, to express basic properties like “the sum of two odd numbers is an even number”.

Representing Integers

- How do we represent integers in first-order logic?
  - Define 0, a constant.
  - Define a successor function.
    1 = successor(0)
    2 = successor(successor(0))
  - We do not have to explicitly define constants 1, 2, and so on. They are implicitly defined as return values of the successor function.
  - We can define predicates such as odd(number), even(number), prime(number) and so on.
- How can we specify which numbers are odd and which numbers are even, with finitely many statements?

Representing Integers

- How can we specify which numbers are odd and which numbers are even, with finitely many statements?

  even(0)
  ∀ x: even(x) => odd(successor(x))
  ∀ x: odd(x) => even(successor(x))

- These three statements describe for infinitely many integers whether they are even or odd.
- Contrast that to propositional logic, where you would need infinitely many symbols and rules.

Adjacency in the Wumpus World

- For the wumpus world, to say that "pits cause breezes in adjacent squares" using first-order logic, we use this rule:

  $\forall$ x1, y1: Breeze(x1, y1) <=>

  $\exists$ x2, y2: Pit(x2, y2) AND Adjacent(x1, y1, x2, y2)

- How can we define the Adjacent relation?

Adjacency in the Wumpus World

- For the wumpus world, to say that "pits cause breezes in adjacent squares" using first-order logic, we use this rule:

  ∀ x1, y1: Breeze(x1, y1) <=>
  ∃ x2, y2: Pit(x2, y2) AND Adjacent(x1, y1, x2, y2)

- How can we define the Adjacent relation?

- ∀ x1, y1: Adjacent(x1, y1, successor(x1), y1)
- ∀ x1, y1: Adjacent(x1, y1, x1, successor(y1))
- ∀ x1, y1, x2, y2: Adjacent(x1, y1, x2, y2) <=>
  Adjacent(x2, y2, x1, y1)

Examples from the Textbook

- How do we say that brothers are siblings?

Examples from the Textbook

- How do we say that brothers are siblings?

  $\forall$ x, y : Brothers(x, y) => Siblings(x, y)

Examples from the Textbook

- How do we say that “siblings” is a symmetric relation?

Examples from the Textbook

- How do we say that “siblings” is a symmetric relation?

  $\forall$ x, y : Siblings(x, y) => Siblings(y, x)

Examples from the Textbook

- How do we say that one’s mother is one’s female parent?

Examples from the Textbook

- How do we say that one's mother is one's female parent?

  $\forall$ x, y : (x = Mother(y)) => (Female(x) $\wedge$ Parent(x, y))

- Alternative: making **Mother** a relation.

  $\forall$ x, y : Mother(x, y) => (Female(x) $\wedge$ Parent(x, y))

Examples from the Textbook

- How do we say that a first cousin is a child of a parents' sibling?

Examples from the Textbook

- How do we say that a first cousin is a child of a parents' sibling?

  $\forall$ x, y : FirstCousin(x, y) <=>

  $\exists$ p, ps : Parent(p,x) $\wedge$ Sibling(ps,p) $\wedge$ Parent(ps,y)

First-Order Logic Syntax

- What is the simplest possible sentence in first-order logic?

First-Order Logic Syntax

- What is the simplest possible sentence in first-order logic?
  - A predicate applied to constants:

    predicate(constant$_1$, … constant$_n$)

- Examples:

First-Order Logic Syntax

- What is the simplest possible sentence in first-order logic?
  - A predicate applied to constants:

    predicate(constant$_1$, … constant$_n$)

- Examples:

Siblings(John, Mary)

>(5, 3)
White(cloud)

First-Order Logic Syntax

- Are these valid first-order logic sentences?

1

John

Mother(Mary)

Mother(Father(Mother(Mary)))

First-Order Logic Syntax

- Are these valid first-order logic sentences?

1

John

Mother(Mary)

Mother(Father(Mother(Mary)))

- No. Each of these lines refers to an object.
- In first-order logic you can refer to an object in three ways:
  - Using a constant, like 1, John.
  - Using a variable (that has been introduced using a quantifier).
  - Using a function call, like Mother(Mary), or more complicated, nested calls, like Mother(Father(Mother(Mary)))

First-Order Logic Syntax

- Objects are an important part of first-order logic.
- However, an object by itself cannot be a first-order logic sentence.
  - A sentence must have a boolean value.
- Then, where do objects appear in sentences?

First-Order Logic Syntax

- Objects are an important part of first-order logic.
- However, an object by itself cannot be a first-order logic sentence.
  - A sentence must have a boolean value.
- Then, where do objects appear in sentences?
- Objects (constants, variables, function calls) appear:
  - As arguments to predicates.
  - On the left and right side of an equal sign.

First-Order Logic Syntax

- At the top level, a first-order logic sentence must be one of the following:
  - An application of a predicate.

    Example: cousins(mother(Mary), father(father(John)))

  - An equality test.

    Example: mother(Mary) = Jane

  - An application of an existential or universal quantifier.

    Example: ∀ x, y : Mother(x, y) => (Female(x) ∧ Parent(x, y))

  - An application of a connective to combine simpler sentences.

    Example: Female(Jane) ∧ Parent(Jane, Edward)

Number of Possible Worlds

- In propositional logic, suppose that you have 100 symbols. How many possible worlds do you have?

Number of Possible Worlds

- In propositional logic, suppose that you have 100 symbols. How many possible worlds do you have?
  - 2^{100}. One for each row of the truth table.

Number of Possible Worlds

- In propositional logic, suppose that you have 100 symbols. How many possible worlds do you have?
  - 2^{100}. One for each row of the truth table.
- In first-order logic, how can we even count the number of possible worlds?
- It is more complicated.

Number of Possible Worlds

- Suppose that we have:
  - Five constants.
  - No functions.
  - One predicate, that takes one argument.
- How many possible worlds can we have?

Number of Possible Worlds

- Suppose that we have:
  - Five constants.
  - No functions.
  - One predicate, that takes one argument.
- How many possible worlds can we have?
- For each possible argument of the predicate, we must specify if the predicate returns true or false.
- We have five possible arguments.
- In total, 2^5 possible worlds.

Number of Possible Worlds

- Suppose that we have:
  - Five constants.
  - No functions.
  - One predicate, that takes **two** arguments.
- How many possible worlds can we have?

Number of Possible Worlds

- Suppose that we have:
  - Five constants.
  - No functions.
  - One predicate, that takes **two** arguments.
- How many possible worlds can we have?
- For each possible combination of arguments of the predicate, we must specify if the predicate returns true or false.
- We have 25 possible combinations of arguments.
- In total, 2^{25} possible worlds.

Number of Possible Worlds

- Suppose that we have:
  - Five constants.
  - No functions.
  - Three predicates, that take <u>**two**</u> arguments.
  - One predicate that takes one argument.
- How many possible worlds can we have?

Number of Possible Worlds

- Suppose that we have:
  - Five constants.
  - No functions.
  - Three predicates, that take **two** arguments.
  - One predicate that takes one argument.
- How many possible worlds can we have?
- For each possible combination of arguments of each predicate, we must specify if the predicate returns true or false.
- For two arguments, we have 25 combinations of arguments. We have three such predicates, so we must specify 75 values.
- For one argument, we have 5 combinations of arguments. We have one such predicate, so we must specify 5 values.
- In total, $2^{5+75} = 2^{80}$ possible worlds.

Inference via Propositionalization

- Suppose that we have:
  - Five constants.
  - No functions.
  - Three predicates, that take **<u>two</u>** arguments.
  - One predicate that takes one argument.
- In such cases, we can automatically convert our first-order logic knowledge base to an equivalent propositional logic knowledgebase.
- For each possible combination of arguments of each predicate, we must define a symbol in the propositional logic version.
- We have 80 total predicate values to specify, so we would need a propositional logic knowledge base with 80 symbols.
- Then, we can do inference using TT-Entails.

Inference via Propositionalization

- Inference via propositionalization can be an attractive option.
- For example, in the wumpus world, we can use first-order logic for our knowledge base, so that we do not have to write 16 different versions of every rule.
- Then, our software can translate our knowledge base to propositional logic, and use TT-Entails.
- This way, we get the elegance of first-order logic, and the finite-time inference algorithms of propositional logic.

Propositionalization: An Example

- Suppose we have two constants: Edward and Mary.
- Suppose that we have two predicates:
  - Tall, takes one argument.
  - Parent, takes two arguments.
- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- What (and how many) symbols do we have to define to propositionalize this knowledge base?

Propositionalization: An Example

- Suppose we have two constants: Edward and Mary.
- Suppose that we have two predicates:
  - Tall, takes one argument.
  - Parent, takes two arguments.
- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- What (and how many) symbols do we have to define to propositionalize this knowledge base?
- 6 symbols:
  - Tall_Edward, Tall_Mary.
  - Parent_Edward_Edward, Parent_Edward_Mary, Parent_Mary_Edward, Parent_Mary_Mary.

Propositionalization: An Example

- Suppose we have two constants: Edward and Mary.
- Suppose that we have two predicates:
  - Tall, takes one argument.
  - Parent, takes two arguments.
- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- How do we translate the knowledge base to propositional logic?

Propositionalization: An Example

- Suppose we have two constants: Edward and Mary.
- Suppose that we have two predicates:
  - Tall, takes one argument.
  - Parent, takes two arguments.
- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- How do we translate the knowledge base to propositional logic?

  Tall_Edward ∧ not(Tall_Mary)
  not(Parent_Edward_Mary)

Propositionalization: An Example

- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- How do we translate the knowledge base to propositional logic?

  Tall_Edward ∧ not(Tall_Mary)
  not(Parent_Edward_Mary)

- Note that the knowledge base does not use all six symbols.
- For example, Parent_Mary_Edward is not used.
- Then, why do we need to have such a symbol in the truth table?

Propositionalization: Symbols Needed

- When we propositionalize a first-order logic knowledge base, we need symbols in the truth table not only for the knowledge base, but also for every possible statement we can form using the constants, predicates, and functions that are used.
- Why? Because for any possible statement S, we may want to check if the knowledge base entails that statement S.
- In order for the propositional logic translation to be equivalent to the first-order logic knowledge base, it should allow us to ask any question that we can ask using first-order logic.
- Parent_Mary_Edward is not needed in the knowledge base, but we may want to ask if the knowledge base entails Parent_Mary_Edward.

Propositionalization Limitations

- Suppose we have two constants: Edward and Mary.
- Suppose that we have one predicate:
  - Tall, takes one argument.
- Suppose that we have one function.
  - Mother, takes one argument.
- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- What (and how many) symbols do we have to define to propositionalize this knowledge base?

Propositionalization Limitations

- Suppose we have two constants: Edward and Mary.
- Suppose that we have one predicate:
  - Tall, takes one argument.
- Suppose that we have one function.
  - Mother, takes one argument.
- Suppose we have this knowledge base:

  Tall(Edward) ∧ not(Tall(Mary))
  not(Parent(Edward, Mary))

- What (and how many) symbols do we have to define to propositionalize this knowledge base?
- Infinitely many symbols!!!

Propositionalization Limitations

- Suppose we have two constants: Edward and Mary.
- Suppose that we have one predicate:
  - Tall, takes one argument.
- Suppose that we have one function.
  - Mother, takes one argument.
- We need a symbol for Tall(Mary).
- We need a symbol for Tall(Mother(Mary)).
- We need a symbol for Tall(Mother(Mother(Mary))).
- And so on, for ever.

Number of Worlds in First-Order Logic

- In any first order logic domain where you do NOT have functions, you can do propositionalization.
- Thus, the number of possible worlds is the same both in the first-order logic representation and in the propositional logic representation.
- However, if you have even a single function, you cannot do propositionalization.
  - You would need infinitely many symbols.
- Whenever you have even a single function, the number of all possible worlds is INFINITE.

Inference in First-Order Logic

- If you have no functions, you can do inference using:
  - Propositionalization.
  - TT-Entails in the propositional-logic version.
- The complexity of inference in that case is exponential, but finite.

Inference in First-Order Logic

- If you have functions: we will NOT study any inference algorithms for that case in this course.
  - We will be briefly looking at the basics of those methods: <u>**resolution**</u>, and other inference methods.
- What I want you to know:
  - For first-order logic, there is no sound and complete inference algorithm that runs in finite time.
  - There are sound and complete algorithms that:
    - If the knowledge base entails the statement, the algorithms will return **true** in a finite (exponential in the worst case) amount of time.
    - If the knowledge base does NOT entail the statement, the algorithms might return **false** in a finite amount of time, but they might also <u>**never terminate**</u>.

Inference Recap

- Inference takes exponential time (in the worst case) in propositional logic.
- In first order logic, inference takes:
  - Exponential time (in the worst case) if the knowledge base entails the statement.
  - Infinite time (in the worst case) if the knowledge base does not entail the statement.
- The time complexity of inference makes it difficult to solve large problems using propositional and first order logic.
- In many interesting cases (but not all cases), there are alternatives that provide good trade-offs.
  - Languages that are somewhat similar to first-order logic.
  - Inference algorithms that are fast.
- Such an interesting case will be our next topic: planning.