innovate achieve lead

BITS Pilani
Pilani Campus

CS/IS F214
Logic in Computer Science

MODULE: **PREDICATE LOGIC**

Semantics

- Satisfiability and Validity
- Validity is undecidable

Predicate Logic: Satisfiability

- Recall:
  - $\Gamma \models_{\iota}^{M} \phi$

    denotes that ϕ evaluates to TRUE given the set of premises Γ
    - under model M and look-up table ι

- We say a predicate logic formula ϕ is ***satisfiable***
  - if it evaluates to TRUE (without any premises) under *some model* **M** and *some look-up table* ι :
    - i.e.

      $\models_{\iota}^{M} \phi$ for some **M** and some ι

Predicate Logic: Validity

- Recall:
  - $\Gamma \models \phi$

    denotes that ϕ evaluates to TRUE given the set of premises Γ under all models (and all look-up tables).
- We say a predicate logic formula ϕ is ***valid***
  - if it evaluates to TRUE (without any premises) *under all models*: i.e. $\models \phi$

Predicate Logic: Validity: Example 1

- **∀X (p(X) --> q(X)) |= (∀X p(X)) --> (∀X q(X))**
- Justification:
  - Let **M** *be any model such that*
    - **∀X (p(X) --> q(X)) $|=^M$ (∀X p(X)) --> (∀X q(X))**
  - Suppose that *for not every element* of (the universe in) M, ***p(X)** is true*:
    - then we are done. (Why?)
  - Otherwise suppose **∀X p(X)**. Therefore, *for every element **a** of (the universe in) **M***
    - **p(a)** is true; then given our premise, **q(a)** is true
      - i.e. **(∀X q(X))** is true and so we are done. (Why?)

Predicate Logic: Validity: Example 2

- **$(\forall X\ p(X)) \rightarrow (\forall X\ q(X)) \not\models \forall X\ (p(X) \rightarrow q(X))$**
  - What does this mean?
    - There is at least one model M in which:
      - **$(\forall X\ p(X)) \rightarrow (\forall X\ q(X))$** is true but
      - **$\forall X\ (p(X) \rightarrow q(X))$** is false.
- Proof:
  - Choose a model M in which **$\forall X\ p(X)$** is not true
    - i.e. $p_M \subset A$, where A is the universe in M
  - Therefore, **$\forall X\ p(X) \rightarrow (\forall X\ q(X))$** is true.
    - Now if q_M and p_M are disjoint
    - then there exists v in A such that
      - p(v) is true but q(v) is not i.e. p(v) --> q(v) is false

Computing Validity

- Recall:
  - We have seen algorithms for computing validity of a formula in propositional logic:
    - *if the formula is in CNF* then validity can be *computed efficiently*
    - *otherwise* it takes *exponential time*.

- Question:
  - Is it there an algorithm to compute ***validity of a predicate logic formula***?

Computing Validity

- To verify that a predicate logic formula is valid:
  - we have to evaluate it in all possible models:
    - *There may be an infinite number of applicable models*.
    - And if a quantifier is involved
      - *the formula may have to be evaluated for all values that a variable can take*.

- Can it not be done in any other way?

Undecidability of Validity in Predicate Logic

- Theorem:
  - ***Validity of formulas in Predicate Logic is undecidable.***
- Recall:
  - A decidable problem is a decision problem that is computable.
    - Computability refers to Turing-computability.
  - *Halting problem is undecidable* i.e.
    - it is not possible to write an algorithm to decide *whether or not a given algorithm / program will terminate on a given input*.

Proving Undecidability

- One approach to proving *that a problem is undecidable* is to start with *a known undecidable problem*:
  - consider a problem **π1**
  - suppose if **π1** can be reduced to another problem **π2** i.e. *there is a mapping **f** on inputs **x** such that*

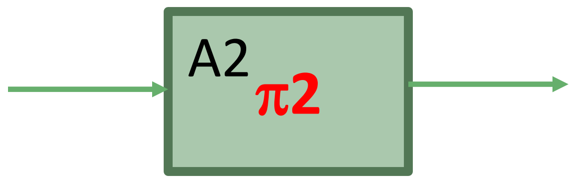
    π1(x) iff π2(f(x)) for all x

    - what does it imply (for **π2**) ?

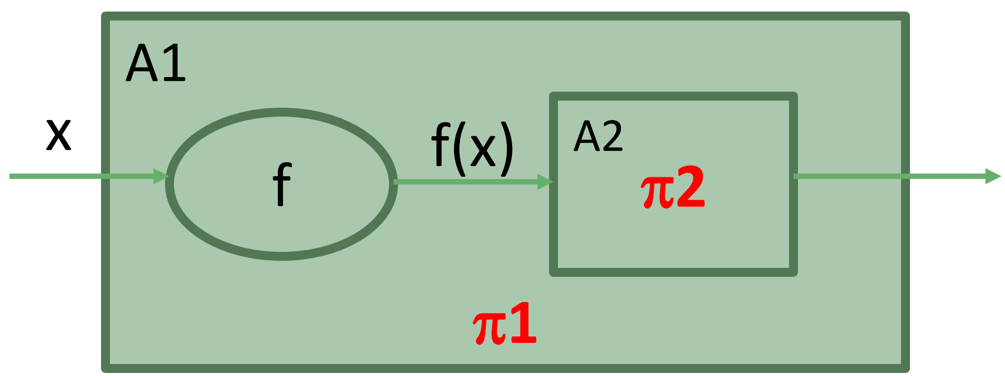
Reduction

- A ***reduction*** from a decision problem π**1** to a decision problem π**2** is
  - a mapping **f** of the inputs of π**1** to the inputs of π**2** such that
    - **π1(x)** is TRUE iff **π2(f(x))** is TRUE for all inputs **x**

- In algorithmic terms, *if there is an algorithm* A2 *for solving* π**2** **:**



- *then one can construct an algorithm* A1 *for* π**1**:



- Assumption:
  - *There is an algorithm for* **f**.

Reduction Theorem

- Reduction Theorem:
  - If an undecidable problem π**1** can be reduced to a problem π**2** then π**2** is undecidable.
- Why is the reduction theorem true?
  - Proof by contradiction:
    - Assume π**2** is decidable, i.e. there is an algorithm to decide π**2**
    - Then use the algorithm for π**2** to construct one for π**1**.
      - But π**1** is undecidable i.e. no such algorithm exists i.e. assumption is wrong!
- Are there any conditions that apply on the reduction?

Undecidability of *Validity in Predicate Logic*

[*approach taken by the text book*]

- Theorem:
  - ***Validity in Predicate Logic* is undecidable**
- Proof:
  - Post's problem is undecidable.
    - [*Proof of this is out of the scope of this course.*]
  - Post's problem reduces to Validity in Predicate Logic.
    - *Refer to the text book for a proof.*
  - Therefore by Reduction Theorem in the previous slide, ***Validity is undecidable***.