Knowledge Representation and Reasoning

Geraint A. Wiggins
Professor of Computational Creativity
Department of Computer Science
Vrije Universiteit Brussel

Objectives

Queen Mary
University of London

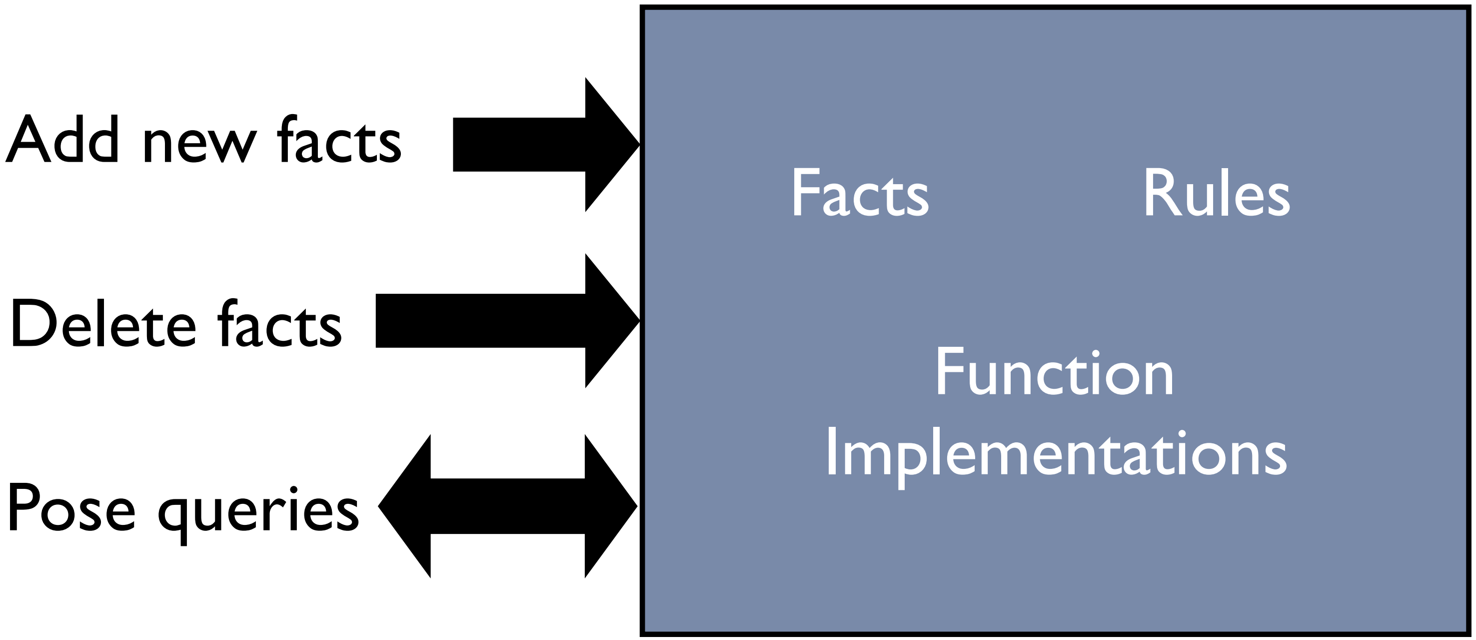
- Knowledge Representation in Logic
  - The Propositional Calculus
  - The First Order Predicate Calculus
- Reasoning
  - Inference Rules to Compute with Calculus Expressions
- Application

Knowledge Engineering

- The role of the Knowledge Engineer is to
  - elicit or otherwise ascertain knowledge
  - represent it in the most appropriate way
  - use it to derive previously unknown facts
    - follow a chain of reasoning from new data to a conclusion (e.g. medical diagnosis)
    - make explicit things that were previously implicit in a system that was too complex for a human to understand all at once
- Examples about VUB site map
  - Building-M-is-a-building
  - Building(M)
  - Grey(M)
  - Colour(M, Grey)

Knowledge Engineering

- Often, in one formalism or another, this will involve maintaining a database of facts that are known to be true and rules that can apply to them



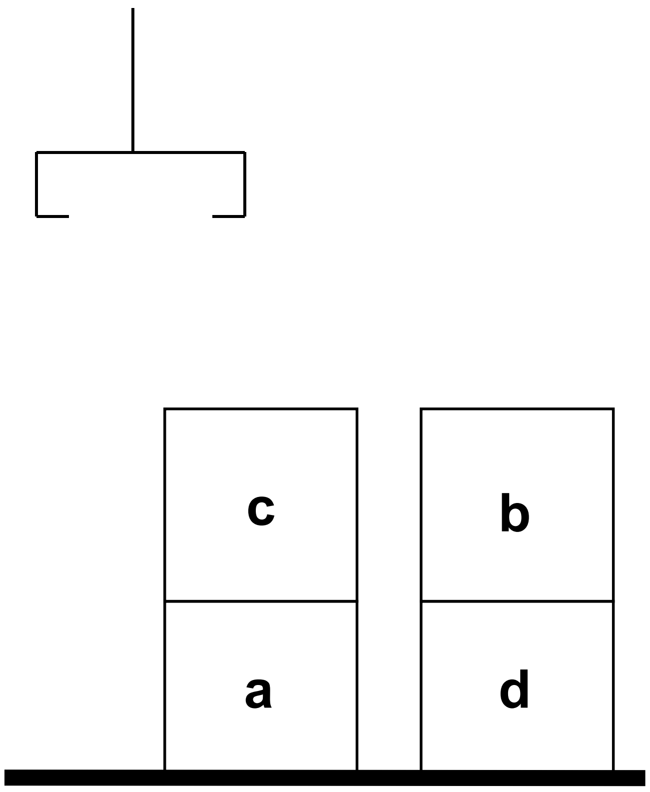
Knowledge Engineering

- Quite often, problem formulation in real-world situations is very difficult
  - different experts have different opinions
  - the world is continuous and unpredictable
  - clients don't really know what they want from you
- A common approach to understanding the issues involved in KE is to use a highly simplified world, and then to generalise with experience
  - a common simplification is the "blocks" world

Example: The Blocks World

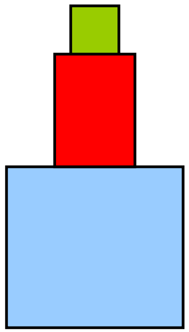
Queen Mary University of London

- There is/are
  - a table
  - some distinguishable blocks
  - a robot hand/arm
- Problems are specified in terms of
  - actions by the arm
  - with respect to the world
- Predicates
  - On, On-table
  - Clear
  - Empty



Knowledge Representation and Inference

- KR should allow us, for a given world, to:
  - Express facts or beliefs using a formal language
    - expressively and unambiguously
- The inference procedure should allow us to:
  - Determine automatically what follows from these facts
    - correctly (sound) and completely (and tractably)
- Example:
  - Be able to express formally that:
    - "The red block is above the blue block"
    - "The green block is above the red block"
  - Be able to infer:
    - "The green block is above the blue block"
    - "The blocks form a tower"



Example

- Given
  - If it is sunny today, then the sun shines on the screen
  - If the sun shines on the screen, then the blinds are drawn
  - The blinds are not down
- Find out
  - Is it sunny today?
- Human reasoning:
  - Blinds up, so sun not shining on screen, so not sunny today
    - We want a computer to do that, reliably and in general

Components of a logical calculus

- A *formal language*
  - words and *syntactic* rules that tell us how to build up sentences
    - so we can build up more complex statements from simple ones
  - *semantic* mappings that tell us what the words mean
- An *inference procedure* which allows us to compute which sentences are valid *inferences* from other sentences
- Many different logical calculi; here we study
  - The Propositional Calculus
  - The First Order Predicate Calculus

The Propositional Calculus

- Each symbol in the Propositional Calculus is
  - a *proposition*: a basic, smallest unit of meaning in the calculus
    - e.g. "it is raining"
  - a *connective*: something combines propositions into more complex *sentences*
- Two reserved, special propositions
  - True and False
    - with the obvious meanings!
- Other propositions usually begun by upper case letters
  - P, Q, Sunny, etc.
- Connectives use special symbols
  - $\wedge$ (and) , $\vee$ (or) , $\neg$ (not), $\rightarrow$ (implies), $\equiv$ (is equivalent to)

Sentences in the propositional calculus

- The Sentence is the syntactic unit to which truth values can be attached
    - Sentences are also called *Well-Formed Formulae (WFF)*
  - Every propositional symbol is a sentence. E.g.: True, False, P
  - The negation of a sentence is a sentence. E.g.: $\neg P$, $\neg$ False.
  - The conjunction (and) of two sentences is a sentence. E.g.: $P \wedge Q$
  - The disjunction (or) of two sentences is a sentence. E.g.: $P \vee Q$
  - The implication of one sentence by another is a sentence. E.g.: $P \rightarrow Q$
  - The equivalence of two sentences is a sentence. E.g.: $P \equiv R$
    - Note that equivalence can also be expressed as $P \rightarrow Q \wedge Q \rightarrow P$
    - $\equiv$ is therefore sometimes omitted from the propositional calculus

Semantics (Meaning) in PC

- An interpretation of a set of sentences is the assignment of a truth value, either T or F, to each propositional symbol (and so to each sentence)
  - The proposition True is always assigned truth value T
  - The proposition False is always assigned truth value F
  - The assignment of negation, $\neg P$, is F iff the assignment of P is T, and vice versa
  - The assignment of conjunction, $P \wedge Q$, is T iff the assignment of both P and Q is T; otherwise it is F
  - The assignment of disjunction, $P \vee Q$, is F iff the assignment of both P and Q is F; otherwise it is T
  - The assignment of implication, $P \rightarrow Q$, is F iff the assignment of P is T and the assignment of Q is F; otherwise it is T
  - The assignment of equivalence, $P \equiv Q$, is T iff the assignments of both P and Q is the same for all possible interpretations; otherwise it is F.

Some useful laws and equivalences

- excluded middle: $P \vee \neg P$
- $\neg \neg P \equiv P$
- contrapositive: $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$
- de Morgan's laws
  - $\neg (P \vee Q) \equiv \neg P \wedge \neg Q$
  - $\neg (P \wedge Q) \equiv \neg P \vee \neg Q$

Some useful laws and equivalences

- excluded middle: $P \vee \neg P$
- $\neg \neg P \equiv P$
- contrapositive: $P \rightarrow Q \equiv \neg Q \rightarrow \neg P$
- de Morgan's laws
  - $\neg (P \vee Q) \equiv \neg P \wedge \neg Q$
  - $\neg (P \wedge Q) \equiv \neg P \vee \neg Q$
- commutativity
  - $P \vee Q \equiv Q \vee P$
  - $P \wedge Q \equiv Q \wedge P$

Some useful laws and equivalences

- associativity
  - $(P \vee Q) \vee R \equiv P \vee (Q \vee R)$
  - $(P \wedge Q) \wedge R \equiv P \wedge (Q \wedge R)$
- distributivity
  - $P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$
  - $P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$
- Note order of *operator precedence*
  - $\neg$ precedes $\wedge$ precedes $\vee$
  - $\rightarrow$ are $\equiv$ are complicated: use brackets
  - Compare with arithmetic operators, –, x, +

Truth tables

- A truth table has all sentences along its top, usually in increasing order of syntactic complexity
  - its columns are all the possible interpretations, one row each

P	**Q**	**¬P**	**P ∧ Q**	**P ∨ Q**	**P → Q**
T	T	F	T	T	T
T	F	F	F	T	F
F	T	T	F	T	T
F	F	T	F	F	T

Truth tables

- We can prove things using truth tables
  - $\neg P \vee Q \equiv P \rightarrow Q$

P	**Q**	**¬P**	**¬P ∨ Q**	**P→Q**	**¬P ∨ Q ≡ P→Q**
T	T	F	T	T	T
T	F	F	F	F	T
F	T	T	T	T	T
F	F	T	T	T	T

Proofs in propositional calculus

- Problem Description
  - If it is sunny today, then the sun shines on the screen.
  - If the sun shines on the screen, then the blinds are drawn.
  - The blinds are not drawn.
  - Is it sunny today?
- Propositions
  - P: It is sunny today.
  - Q: The sun shines on the screen.
  - R: The blinds are down.
- Premises
  - $P \rightarrow Q$
  - $Q \rightarrow R$
  - $\neg R$
- Question: P? (Given Premises are true, is P true?)

Proof using a truth table

Queen Mary
University of London

Propositions			Premises			Trial conclusions	
P	**Q**	**R**	**P→Q**	**Q→R**	**¬R**	**P**	**¬P**
T	T	T	T	T	F	T	F
T	T	F	T	F	T	T	F
T	F	T	F	T	F	T	F
T	F	F	F	T	T	T	F
F	T	T	T	T	F	F	T
F	T	F	T	F	T	F	T
F	F	T	T	T	F	F	T
F	F	F	T	T	T	F	T

- When all the premises are true, P is false, so “it is not sunny”

First Order Predicate Calculus

- The Propositional Calculus is not very expressive
  - can’t make statements about all of a certain thing
  - or about things that don’t exist
  - or about whether things exist
  - and we can’t give propositions whole interpretation depends on which thing they apply to
- In the “blinds” example, we had to omit the day on which we checked the premises
- How could we make statements to capture the idea that we’d do this procedure each day?
  - If it is sunny on Monday ...
  - If it is sunny on Tuesday ... etc.

First Order Predicate Calculus

Queen Mary
University of London

- The First Order Predicate Calculus (FOPC) is a *conservative extension* of the Propositional Calculus (PC)
  - this means that it has all the properties and features of PC
  - and some extra ones
    - objects: things which sentences are about, written like propositions
    - variables: usually written as lower case single letters, ranging over objects
    - predicates: propositions are now predicate symbols which can apply to variables and objects, written like propositions
    - arguments: the variables or objects to which predicates and functions apply
    - functions: mappings between objects objects
    - quantifiers: existential $\exists$ - "there exists"; universal $\forall$ - "for all"
- In PC, propositions were predicates that had no arguments

Blinds example in FOPC

- Problem description
  - If it is sunny [on a particular day], then the sun shines on the PC screen [on that day].
  - If the sun shines on the PC screen [on a particular day], the blinds are down [on that day].
  - The blinds are not down [today].
  - Is it sunny [today]?
- Premises:
  - $\forall d.\text{Sunny}(d) \rightarrow \text{Screen-shines}(d)$
  - $\forall d.\text{Screen-shines}(d) \rightarrow \text{Blinds-down}(d)$
  - $\neg\text{Blinds-down}(\text{Thursday})$
- Question: Sunny(Thursday)?
- Note that there are various similar notations for quantifiers

Functions in FOPC

Queen Mary University of London

- A function maps its arguments to a fixed single value
  - note that functions do not have truth values: they map between objects
  - functions are denoted in the same way as predicates
    - you can tell which is which from where they appear: Predicates are outermost
  - functions have an *arity*: the number of arguments they take
- A person's mother is that person's parent
  - $\forall x.\mathrm{Person}(\ x\ ) \rightarrow \mathrm{Parent}(\ \text{Mother-of}(\ x\ ), x\ )$
    - Note that a person can only have one mother, so using a function like this is OK
- There is at least one person in this class who thinks
  - $\exists x.\mathrm{Person}(\ x\ ) \wedge \mathrm{Class}(\ x, \mathrm{AIClass}\ ) \wedge \mathrm{Thinks}(\ x\ )$
- All computers have a mouse connected by USB
  - $\forall x.\mathrm{Computer}(\ x\ ) \rightarrow \exists y.\mathrm{Mouse}(\ y\ ) \wedge \mathrm{Connected}(\ x, y, \mathrm{Usb}\ )$

Syntax of FOPC

Queen Mary University of London

- Terms: corresponding with things in the world
  - Objects
    - e.g., Thursday
  - Variables
    - e.g., x
  - Function expressions
    - A function symbol of arity n followed by n terms, enclosed in () and separated by ,
    - e.g., Function(var, AnotherFunction(Thing))
- Sentences: statements that can be true or false
  - Atomic Sentence
    - A predicate symbol of arity n followed by n terms, enclosed in () and separated by ,
    - Note that n can be 0, so True and False are atomic sentences
  - The result of applying a connective (as in PC) to one or more sentences
  - The result of applying a quantifier (∀, ∃) to a sentence

Semantics of FOPC: Interpretation

Queen Mary
University of London

- Let the *domain* D be a nonempty set of constants, variables, predicate symbols, function symbols and their mappings
- An *interpretation* over D is an *assignment* of the entities in D to each of the constant, variable, predicate, and function symbols of a predicate calculus expression
  - Each constant is assigned an element of D
  - Each variable is assigned to a nonempty subset of D (*allowable substitutions*)
  - Each function of arity m is defined ($D^m \longmapsto D$)
  - Each predicate of arity n is defined (D^n to {T,F}).

Computing the truth value of predicate calculus sentences

- Given an expression E and an interpretation I of E over a nonempty domain D, the truth value for E is determined by
  - The value of a constant is the element of D it is assigned to by I
  - The value of a variable is a member of the set of elements of D it is assigned to by I
  - The value of a function expression is the element of D obtained by evaluating the function for the parameter values assigned by the interpretation
  - The value of the predicate "true" is T, and the predicate "false" is F
  - The value of an atomic sentence is either T or F, determined by I
  - The value of a non-atomic (compound) sentence is either T or F, determined by I
  - For a variable x and a sentence S containing x
    - The value of $\forall x.S$ is T if S is T for all assignments to x under I
    - The value of $\exists x.S$ is T if there is an assignment to x under I such that S is T

First, Second and Higher Order PC

- This is First Order Predicate Calculus
  - variables can range only over objects in D
    - John eats everything: $\forall x.Eats(John, x)$
- In Second Order Predicate Calculus
  - variables can range over objects, predicates and functions in D
    - John has all the features that Jim has: $\forall P.P(Jim) \rightarrow P(John)$
- In Higher Order Predicate Calculus
  - variables can range over objects, predicates, functions in D and over sentences
- In this module, we consider only First Order Predicate Calculus

Order and range of quantifiers matters

- Every person likes some food
  - $\forall x. \text{Person}(x) \rightarrow \exists f. \text{Food}(f) \wedge \text{likes}(x, f)$
- There is a food that every person likes
  - $\exists f. \text{Food}(f) \wedge \forall x. \text{Person}(x) \rightarrow \text{Likes}(x, f)$
- Whenever anyone eats some spicy food, they are happy
  - $\forall x. \exists f. \text{Eats}(x, f) \wedge \text{Spicy}(f) \rightarrow \text{Happy}(x)$
    - allowable substitutions for x are people, for f is food
  - $\forall x. \text{Person}(x) \rightarrow \exists f. \text{Food}(f) \wedge \text{Spicy}(f) \wedge \text{Eats}(x, f) \rightarrow \text{Happy}(x)$
    - no need to worry about allowable substitutions

Equality

- A very useful extra operator that isn't strictly in FOPC is =
  - that's to say, the TEST for equality, like == in Java, not assignment
- The rule for = is that
  - A = A is true for all constants A in the interpretation
  - otherwise, it is false
- We'll use equality in some of our lab work

Domains

Queen Mary
University of London

Syntax

Constant names: Edna Fido Park
Predicate names: DogWalk/3

DogWalk(Edna, Fido, Park)

Constant names: Mehwesh Gym Weights
Predicate names: In/2 Lifts/2

Lifts(Mehwesh, Weights) ∧
In(Mehwesh, Gym)

Constant names: Dave Sea
Predicate names: On/2 Boat/1

∃b.Boat(b) ∧
On(Dave, b) ∧
On(b, Sea)

Semantic Domain

Objects: Edna Boat Dave Fido Gym Mehwesh Park Sea Weights

Predicates: Boat/1 DogWalk/3 In/2 Lifts/2 On/2

Interpretation

DogWalk(Edna, Fido, Park)	T
DogWalk(Dave, Fido, Park)	F
Boat(Boat)	T
Boat(Fido)	F

etc.

World

Something to try

- John's meals are spicy

- Every city has a dogcatcher who has been bitten by every dog in town
  - With domains – what are they?

  - Without domains

Something to try

- John's meals are spicy
    - $\forall x.\ \text{Meal-of}(\text{John}, x) \rightarrow \text{Spicy}(x)$
- Every city has a dogcatcher who has been bitten by every dog in town
    - With domains – what are they?
        - $\forall c. \exists t. \text{Dogcatcher}(c, t) \wedge \forall d. \text{Lives-in}(d, c) \rightarrow \text{Has-bitten}(d, t)$
        - Domains: t: people; c: cities; d: dogs
    - Without domains
        - $\forall c. \text{City}(c) \rightarrow \exists t. \text{Dogcatcher}(c, t) \wedge \forall d. \text{Dog}(d) \wedge \text{Lives-in}(d, c) \rightarrow \text{Has-bitten}(d, t)$

Something to try

- For every set x, there is a set y, such that the cardinality of y is greater than the cardinality of x
  - With domains – what are they?
  - Without domains

Something to try

- For every set x, there is a set y, such that the cardinality of y is greater than the cardinality of x
  - With domains – what are they?
    - $\forall x. \exists y. \forall u. \forall v. \text{Cardinality}(x, u) \land \text{Cardinality}(y, v) \rightarrow \text{Greater-than}(v, u)$
    - $\forall x. \exists y. \text{Greater-than}(\text{Cardinality}(x), \text{Cardinality}(y))$
    - Domains: x, y: sets; u,v: integers
  - Without domains
    - $\forall x. \text{Set}(x) \rightarrow \exists y. \text{Set}(y) \land \forall u. \forall v. \text{Cardinality}(x, u) \land \text{Cardinality}(y, v) \rightarrow \text{Greater-than}(v, u)$
    - $\forall x. \text{Set}(x) \rightarrow \exists y. \text{Set}(y) \land \text{Greater-than}(\text{Cardinality}(x), \text{Cardinality}(y))$

Properties of sentences

- For a predicate calculus sentence, S, and an interpretation, I,
  - I *satisfies* S, if S has a truth value of T under I and at least one variable assignment
  - I *is a model of* S, if I satisfies S for all possible variable assignments in I
- A sentence is *satisfiable* iff there is at least one interpretation and variable assignment that satisfy it; otherwise it is *unsatisfiable*
- A set of sentences, E, is *satisfiable* iff there is at least one interpretation and variable assignment that satisfies every $S \in E$
  - NB quantification! The same interpretation/variable assignment pair satisfies all S
- A set of sentences is *inconsistent*, iff it is not satisfiable
- A sentence is *valid* iff it is satisfiable for all possible interpretations

Proof procedures

- A proof procedure consists of
  - a set of inference rules
  - an algorithm for applying the inference rules to a set of sentences to generate a sequence of set of sentences from or to another set
    - usually, we attempt to start from something we want to prove
    - and then work "backwards" to things we already know, such as axioms and theorems
- Semantics of logical entailment
  - A sentence, S, *logically follows from*, or *is entailed by*, a set, E, of sentences iff every interpretation and variable assignment that satisfies E also satisfies S.

Inference Rules

Queen Mary
University of London

- Soundness
  - An set of inference rules is *sound,* iff every sentence it infers from a set, E, of sentences logically follows from E
- Completeness
  - An set of inference rules is *complete,* iff it can infer every expression that logically follows from a set of sentences

Inference Rules

Queen Mary University of London

- Modus Ponens (implication elimination)
  - We know that P implies Q, and that P is true, so Q is true
  - $(P \land (P \to Q)) \to Q$

$$\frac{P \quad P \to Q}{Q}$$

- Modus Tollens
  - We know that P implies Q, and that Q is false, so P is false
  - $(\neg Q \land (P \to Q)) \to \neg P$

$$\frac{\neg Q \quad P \to Q}{\neg P}$$

- But we need rules to deal with each connective
  - Introduction (adding a connective into a proof sequence)
  - Elimination (removing a connective from a proof sequence)

Inference rules

- Conjunction (And) elimination
  - P is true and Q is true if P ∧ Q is true

$$\frac{P \wedge Q}{P \qquad Q}$$

- Conjunction (And) introduction
  - P ∧ Q is true if P is true and Q is true

$$\frac{P \qquad Q}{P \wedge Q}$$

- Universal (Forall) elimination
  - P(a) is true for all constants, a, if ∀x.P(x) is true

$$\frac{\forall x.P(x)}{P(a)}$$

Inference rules

- Conjunction (And) elimination
  - P is true and Q is true if $P \wedge Q$ is true

$$\frac{P \wedge Q}{P \qquad Q}$$

- Conjunction (And) introduction
  - $P \wedge Q$ is true if P is true and Q is true

$$\frac{P \qquad Q}{P \wedge Q}$$

- Universal (Forall) elimination
  - P(a) is true for all constants, a, if $\forall x.P(x)$ is true

$$\frac{\forall x.P(x)}{P(a)}$$

- Universal (Forall) introduction
  - $\forall x.P(x)$ is true, if $P(a_i)$ is true for all constants, a_i

$$\frac{P(a_1) \ldots P(a_n)}{\forall x.P(x)}$$

Blinds example in FOPC revisited

- Problem description
  - If it is sunny [on a particular day], then the sun shines on the screen [on that day].
  - If the sun shines on the screen [on a particular day], the blinds are down [on that day].
  - The blinds are not down [today].
  - Is it sunny [today]?
- Premises:
  - $\forall d.\text{Sunny }(d) \rightarrow \text{Screen-shines}(d)$
  - $\forall d.\text{Screen-shines}(d) \rightarrow \text{Blinds-down}(d)$
  - $\neg\text{Blinds-down}(\text{Thursday})$
- Question: Sunny(Thursday)?

Blinds example in FOPC revisited

Universal instantiation ∀d.Screen-shines(d)→Blinds-down(d)

Modus Tollens ¬ Blinds-down(Thu) Screen-shines(Thu)→Blinds-down(Thu)

∀d.Sunny (d)→Screen-shines(d)

Universal instantiation

Sunny(Thu)→Screen-shines(Thu)

¬ Screen-shines(Thu)

Modus Tollens

¬Sunny(Thu)

- Premises:
  - ∀d.Sunny(d)→Screen-shines(d)
  - ∀d.Screen-shines(d)→Blinds-down(d)
  - ¬Blinds-down(Thursday)
- However, this is quite complicated, and requires knowledge
- Better to be simpler
  - use Resolution Theorem Proving