The FO(·) Knowledge Base System project
Lecture one: The logic framework FO(.)

May 25, 2016

In the KB-paradigm, a logic theory:

- is not a program
- cannot be executed
- cannot be run
- does not do anything
- does not specify a problem

In the KB-paradigm,

- a logic theory is a knowledge base
- it is a bag of information/knowledge.

Knowledge in philosophical logic

What is knowledge?

- True, justified belief of the agent in the real world

What is belief?

- Information/propositions believed to be true by the agent in the real world

But what is information?

Knowledge in KRR & computational logic

- In our applications (ASP, constraint programming), the information we represent is often not true and justified in the real world.
- The domains or systems that we specify are often not the real world:
  - may be for the future (e.g., computing a schedule),
  - maybe only in our imagination (e.g., a game)
- So, a philosphical logician might argue that what we do
  - is not knowledge representation
  - but information representation.

My terminology:

Knowledge Representation
=
formal specification
=
modelling in a logic

We specify "information" about an existing, future or imaginary domain.

What are the most important human resources?

What are the most important human resources?

Air, water, food, . . .

What are the most important human resources?

Air, water, food, . . . information?

What are the most important human resources?

Air, water, food, . . . information?

Where is the formal science of knowledge/information?

- There is no commonly known, well accepted science.

Logic and the scientific study of information

- Logic is usually defined as the study of correct reasoning.
- But, all reasoning needs a resource: this is information.
- And so, the formal study of reasoning (logic) went hand in hand with the development of a formal language to represent information: this is how classical logic came about.
- The old ones were aware of this.

- *Gottfried Wilhelm Leibniz* ($\pm$ 1700) :
  - That complex thoughts in our mind are composed from simpler thoughts, using mathematical operators analogous to arithmetical operators $+, \times, -, \ldots$.
- *George Boole* ($\pm$ 1850):
  - identified a number of these operators : $\wedge, \vee, \neg$ and developed Boolean algebra, aka propositional logic.

    An Investigation of the Laws of Though (1854)
- *Gottlob Frege* ($\pm$ 1880)
  - added quantification a.o. . His Begriffschrift (1879) counts as the first presentation of classical logic.

- According to Alex Bochman (personal communication), the faith in being able to formally study information waned around 1920.
- When Lewis discovered the problems leading up to modal logic ($\pm$1915)
  - "The morning star is the evening star, but it is not a necessarily the case that the morning start is the evening star"

- According to Alex Bochman (personal communication), the faith in being able to formally study information waned around 1920.
- When Lewis discovered the problems leading up to modal logic ($\pm$1915)
  - "The morning star is the evening star, but it is not a necessarily the case that the morning start is the evening star"

Formal science: how does it work?

- A formal science of some empirical domain uses mathematical methods to build abstractions of the entities in the domain.
- What is certainly needed to have a formal science:
  - Mathematical objects that are abstract representations of the real entities in the domain
  - A clear understanding of the link between mathematical objects and the real entities
  - An experimental verification methodology to refute or corroborate mathematical results.

An example – physics, Newtons gravity theory

Newtons gravity theory restricted to two bodies b_1, b_2:

One body's acceleration is proportional to the mass of the other bodiy and inverse proportional to the square of the distance.

$$\frac{d^2Fi(t)}{d^2t}(t) = G \times \frac{m_2}{||F_2(t) - F1(t)||^2}$$
$$\frac{d^2F2(t)}{d^2t}(t) = G \times \frac{m_1}{||F_2(t) - F1(t)||^2}$$

where objects b_i are abstracted as pairs (m_i, F_i) with

- m_i: the mass,
- $F_i : \mathbb{R}^+ \rightarrow \mathbb{R}^3$: the trajectory over time.

- (Mathematical) structures:
  - assignments of values to the symbols m_i, F_i.
  - they are abstractions of potential states of affairs of the empirical reality
- Solutions: structures that satisfy the equations
- This scientific theory represents information about the physical reality by allowing to distinguish between:
  - possible worlds having abstractions that are structures satisfying the theory
  - impossible worlds having abstractions that do not satify the theory
- Calculemus!: A range of very different problems can now be solved.

Newtons gravitation theory is a formal empirical scientific theory

- there are mathematical abstractions, with good understanding of their link to the empirical reality
- there is an experimental methodology
- it was refuted
- but it is still of great utility
  - it is approximately correct in a range of well-understood conditions
  - extremely accurate under standard conditions

Some problems with a formal science of information

- Information is not directly observable
  - it is a thought
  - it resides in our brain
  - it can be expressed only via natural language

I am not here in the happy position of a mineralogist who shows his audience a rock-crystal: I cannot put a *thought* in the hands of my readers with the request that they should examine it from all sides. Something in itself not perceptible by sense, the thought is presented to the reader—and I must be content with that—wrapped up in a perceptible linguistic form.

Gottlob Frege, Der Gedanke

Some problems with a formal science of information

- Information is only observable via natural language
- Natural language is unreliable, vague, ambiguous
- Information and natural language are of overwhelming complexity

It is here that Montague made his biggest contribution:

To most logicians (like the first author) trained in model-theoretic semantics, natural language was an anathema, impossibly vague and incoherent.

To us, the revolutionary idea in Montagues PTQ paper (and earlier papers) is the claim that natural language is not impossibly incoherent, as his teacher Tarski had led us to believe, but that large portions of its semantics can be treated by combining known tools from logic, tools like functions of finite type, the λ-calculus, generalized quantifiers, tense and modal logic, and all the rest.

Jon Barwise, Robin Cooper. 1981. Generalized quantifiers and natural language. Linguistics and Philosophy 4(4). 159219.

Some problems with a formal science of information

- Information is only observable via natural language
- Unreliablity, vagueness, ambiguity of natural language
- Overwhelming complexity of information and of natural language
- Absence of widely accepted mathematical modellings of information.

Current views in Computational Logic and Knowledge Representation

- Natural language is considered impossibly vague, incoherent, ambiguous.
- "Information" is unreachable for the methods of formal science.
- Reflected in the attitude towards informal semantics of knowledge representation logics
  - Informal semantics = general theory of what information is expressed by formal expressions of the logic
  - The link between
    - formal mathematical objects: logic expressions, and
    - the reality that it represents : the information, which may be our knowledge
  - Informal semantics of KR logics: ignored, distrusted, poorly understood, considered of little importance

E.g., in Logic Programming and ASP what is the informal semantics/intuitive meaning of

- negation as failure?
- the rule operator?

In contrast, some formal empirical sciences give extreme attention to the link between

- formal mathematical objects
- the real entities that they represent.

E.g., Consider the methods of extreme accuracy in physics to measure weight, distance, time.

- Measuring is translating empirical phenomena to mathematical abstractions.

Some mitigating observations

On the complexity of information

- Yes, information can be extremely complex in general, even in small talk in NL.

On the complexity of information

- Yes, information can be extremely complex in general, even in small talk in NL.
- But in application domains of declarative problem solving and formal specification, knowledge is simple and objective
  - E.g., a configuration problem.
  - No introspective agents around, no complex propositional attitudes.

Rome was not built on one day. Lets start to study this simple information.

On the ambiguity of natural language

- Yes, natural language can be very complex, vague, incoherent, ambiguous.

On the ambiguity of natural language

- Yes, natural language can be very complex, vague, incoherent, ambiguous.

- But if used with care, it can be very precise and clear too.
- E.g., mathematicians and formal scientists build formal sciences in natural language, and they are trained to convey their thoughts with mathematical precision.
    - E.g., "a prime number is a number divisible only by 1 and itself."
- If natural language could not be precise, formal science could not even be built.

A "quantum of information":

- "Exactly one person in this room wears a hat."

A simple clear piece of objective information.

Small talk:

- "Yesterday, I wanted to go to the sea, but it was bad wheather and I changed my mind."

A sentence with quite clear meaning but of really complex modal nature.

So, let us start by building formal languages based on the precise language constructs used in formal science and mathematics.

Informal semantics

- If we build a formal language based on clear, precise language constructs used in mathematics and formal science, it should be possible to give it a precise informal semantics.

Experimental validation through possible world analysis

- Wittgenstein: if you believe that two propositions in natural language are not equivalent, you should be able to give a situation in which one is true and the other is false.
- $\Rightarrow$ possible world analysis: a simple experimentation method.

Experimental validation through possible world analysis

- Wittgenstein: if you believe that two propositions in natural language are not equivalent, you should be able to give a situation in which one is true and the other is false.
- $\Rightarrow$ possible world analysis: a simple experimentation method.
- Does not work always, but works often and it is surprisingly simple and effective, even in subtle cases.
- In use in philosophical logic.
- I will use it to prove non-equivalence of two different views on the meaning of negation as failure.

E.g., stress in a sentence is important to understand the meaning of the sentence. I.e., stress on a different word may result in a different meaning.

Apply a possible world analysis to show that they are not equivalent:

- “I” did not drive your car.
- I did not drive “your” car.
- I did not drive your “car”.

E.g., stress in a sentence is important to understand the meaning of the sentence. I.e., stress on a different word may result in a different meaning.

Apply a possible world analysis to show that they are not equivalent:

- "I" did not drive your car.
- I did not drive "your" car.
- I did not drive your "car".

E.g., a situation where Bob drove your car, and I did not drive at all, satisfies the first and violates second and third.

Formal model semantics

Formal possible world semantics: (prototype is FO semantics)

- Structures to represent abstractions of potential states of affairs.
- A mathematical theory formalizing the value of formal expressions in structures.

This naturally induces a number of basic semantical concepts:

- the satisfaction/truth relation: $\mathfrak{A} \models \varphi$ if the value of the (boolean) expression φ is $\mathbf{t}$ (true).
  - $\mathfrak{A}$ is a model of φ.
- Per expression, a function from structures to its value: an infon, an intensional object (Montague).
- Entailment relation, validity, tautology, contradiction, satisfiability/consistency.

This simulates how formal science works.

- Remember: solutions of Newtons theory are mathematical structures that satisfy it.

Answer set programming

- In origin, the stable semantics of logic programs (LP) and extended logic programs (ELP) is not a possible world semantics.
    - Gelfond and Lifschitz
- A stable model = a belief set, the set of literals in one epistemic state of a rational introspective agent whose beliefs are expressed by the program.
- ELP was the basis of ASP.

A problem of model semantics

- Possible world analysis and model semantics are under fire.
- It suffices to analyse meaning up to the level of logical equivalence
- There are deeper levels of meaning than logical equivalence.
- E.g., all tautologies are equivalent, but they do not have the same meaning.

Compare (non-tautologies)

$$P \land (P \Rightarrow Q)$$

$$Q \land (Q \Rightarrow P)$$

- Are they logically equivalent?
- Do they have the same meaning?

Yes, but

- In a mature formal empirical science, one frequently uses scientific theories that are only approximately correct.
  - E.g., Newtons gravitation theory
- The point then is to understand where they work correctly, under what conditions they are (sufficiently) precise.
- Formal model theory is sufficiently precise to be extremely useful in formal specification and declarative problem solving.
- So, lets be mature scientists, and use model theory for what it is worth and see where it is not sufficiently precise.

Conclusion

For the FO(.)-KB project:

- Focuss on language constructs known from mathematical text
- Formal model semantics as possible world analysis of mathematical precision
- Clear and precise informal semantics.

In origin, ELP was based on logics from common sense reasoning

- embeddings to default logic
- embeddings to autoepistemic logic

Mathematical information versus common sense information

Why FO as a foundation ?

FO: the language that failed in the seventies?

- Too expressive for building "practical" systems?
  - Undecidability
  - Expressivity/Efficiency trade-off
- FO is not suitable for describing common sense knowledge?
  - Nonmonotonic reasoning
- FO as a language is too difficult for practical use?
  - E.g., quantifiers

Why FO as a foundation ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

Why FO as a foundation ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

- FO is about a small set of connectives:

  $$\wedge, \vee, \neg, \forall, \exists, \Leftrightarrow, \Rightarrow$$

  - Essential for KR, the right semantics in FO

Why FO as a foundation ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

- FO is about a small set of connectives:

  $$\wedge, \vee, \neg, \forall, \exists, \Leftrightarrow, \Rightarrow$$

  - Essential for KR, the right semantics in FO

- Crystal clear informal semantics

  $$\forall x(Human(x) \Rightarrow Man(x) \vee Woman(x))$$

  means

  All humans are men or women

Why FO as a foundation ?

- FO, the outcome of 18's and 19's century's research in

  "laws of thought"

  - E.g., Leibniz, De Morgan, Boole, Frege, Peirce

- FO is about a small set of connectives:

  $$\land, \lor, \neg, \forall, \exists, \Leftrightarrow, \Rightarrow$$

  - Essential for KR, the right semantics in FO

- Crystal clear informal semantics

  $$\forall x(Human(x) \Rightarrow Man(x) \lor Woman(x))$$

  means

  All humans are men or women

- Model semantics as a way to formalize meaning.

Claims

(1) Every expressive declarative modelling language has a substantial overlap with FO, in one form or the other.

Claims

(1) Every expressive declarative modelling language has a substantial overlap with FO, in one form or the other.

- For some languages, the syntax, conceptuology, terminology may severely obscure the relationship.
  - SQL
  - ALLOY
  - Zinc (a constraint programming language)
  - Answer Set Programming

Claims

(1) Every expressive declarative modelling language has a substantial overlap with FO, in one form or the other.

- For some languages, the syntax, conceptuology, terminology may severely obscure the relationship.
  - SQL
  - ALLOY
  - Zinc (a constraint programming language)
  - Answer Set Programming

(2) But FO is not enough for practical KR.

- FO is undecidable?

- FO is undecidable?
- To be precise:
  - Deductive inference in FO is Undecidable
- Other forms of inference are tractable (P) or almost (NP)
- Other forms of inference have far more applications.

- FO is undecidable?
- To be precise:
  - Deductive inference in FO is Undecidable
- Other forms of inference are tractable (P) or almost (NP)
- Other forms of inference have far more applications.
- Much richer “logics” are in use in industry (SQL, ILOG, Zinc)
  - Implementing other forms of inference

- In the FO(·)-KBS project, we focus on knowledge, information
- The aim is to develop expressive natural KR languages suitable to express domain knowledge in real problems.

- In the FO($\cdot$)-KBS project, we focus on knowledge, information
- The aim is to develop expressive natural KR languages suitable to express domain knowledge in real problems.

- (We ignore the expressivity/tractability trade-off in the development of the logic.)
- We hope that cheaper forms of inference will suffice to solve the problems.
- We hope that more expressive KR logics are better in capturing the natural structure of the knowledge, and that this leads to improved efficiency.

Vocabularies and structures in FO

- Vocabulary Σ: a set of constant, predicate and function symbols
- Σ-structures $\mathfrak{A}$
  - a set D, called the domain or the universe of $\mathfrak{A}$;
  - for each symbol $\sigma \in \Sigma$ an appropriate value $\sigma^{\mathfrak{A}}$:
    - for function symbol $(F/n:) \in \Sigma$, a (total) function $F^{\mathfrak{A}} : D^n \to D$;
    - for each predicate symbol $P/n \in \Sigma$, an n-ary relation $P^{\mathfrak{A}} \subseteq D^n$;

We often denote D by $D_{\mathfrak{A}}$.

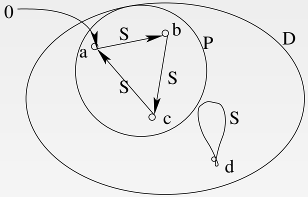
Values of symbols of arity 0:

- for each constant $C \in \Sigma$, $C^{\mathfrak{A}}$ is a domain element of D.
- for each propositional symbol $P \in \Sigma$, $P^{\mathfrak{A}} \in \mathbb{B} = \{\mathbf{t}, \mathbf{f}\}$.

Denoting structures

Different notations for a structure $\mathfrak{A}$ of $\Sigma = \{0/0 :, S/1 :, P : 1\}$.

- Graphical notation:



- Mathematical notation:
  - $D = \{a, b, c, d\}$
    ($a, \ldots, d$ are identifiers for domain elements, not symbols from Σ.)
  - $O^{\mathfrak{A}} = a$
  - $S^{\mathfrak{A}} = \{(a, b), (b, c), (c, a), (d, d)\}$
  - $P^{\mathfrak{A}} = \{a, b, c\}$

Vocabulary and structure in IDP

```
vocabulary V{
    type D
    O:D
    P(D)
    S(D):D
    }
structure S:V{
    D={ a; b; c; d}
    O=a
    S={a-> b; b->c; c->a; d->d}
    P={a; b; c}
}
```

Symbols a,..,d are identifiers/place holders of objects, not constant symbols. They are not allowed to occur in logical formulas.

Structures in IDP

They play an important role in IDP to express domains

- They are used to express basic facts.
- They play the same role as a database

Structures in IDP

They play an important role in IDP to express domains

- They are used to express basic facts.
- They play the same role as a database

Structures in IDP correspond to sets of atoms in ASP.

Syntax in Bachus Naur Form (BNF):

s, t, u, v	$::=$	C	term (constant)
	$\mid$	x	variable
	$\mid$	$f(s_1, \ldots, s_n)$	functional term
A, B	$::=$	$P(s_1, \ldots, s_n)$	atom
	$\mid$	$(s = t)$	equality atom
α, β, φ	$::=$	A	atomic formula
	$\mid$	$\neg\alpha$	negation
	$\mid$	$(\alpha \wedge \beta)$	conjunction
	$\mid$	$(\alpha \vee \beta)$	disjunction
	$\mid$	$(\alpha \Rightarrow \beta)$	material implication
	$\mid$	$(\alpha \Leftrightarrow \beta)$	equivalence
	$\mid$	$\forall x\ \alpha$	universal quantification
	$\mid$	$\exists x\ \alpha$	existential quantification

Interpretation/evaluation of terms

Let $\mathfrak{A}$ interpret all free symbols of term t.

The interpretation $t^{\mathfrak{A}}$ of term t in structure $\mathfrak{A}$ is defined by induction on the structure of t:

- If t is a constant symbol, then $t^{\mathfrak{A}}$ is the domain element assigned by $\mathfrak{A}$ to t.
- If t is a term $F(t_1, \ldots, t_n)$ then $t^{\mathfrak{A}} = F^{\mathfrak{A}}(t_1^{\mathfrak{A}}, \ldots, t_n^{\mathfrak{A}})$.

Sometimes, we call $t^{\mathfrak{A}}$ the value of t in $\mathfrak{A}$.

The satisfaction relation

Let $\mathfrak{A}$ interpret all free symbols of formule φ.

We define that $\mathfrak{A}$ satisfies φ (denoted $\mathfrak{A} \models \varphi$) by induction on the structure of φ:

- $\mathfrak{A} \models P(t_1, \dots, t_n)$ if $(t_1^{\mathfrak{A}}, \dots, t_n^{\mathfrak{A}}) \in P^{\mathfrak{A}}$;
- $\mathfrak{A} \models \alpha \wedge \beta$ if $\mathfrak{A} \models \alpha$ and $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \alpha \vee \beta$ if $\mathfrak{A} \models \alpha$ or $\mathfrak{A} \models \beta$ or both;
- $\mathfrak{A} \models \neg\alpha$ if $\mathfrak{A} \not\models \alpha$; (that is, $\mathfrak{A}$ does not satisfy φ);
- $\mathfrak{A} \models \alpha \Rightarrow \beta$ if it is the case that if $\mathfrak{A} \models \alpha$ then $\mathfrak{A} \models \beta$; i.e., either $\mathfrak{A} \not\models \alpha$ or $\mathfrak{A} \models \beta$
- $\mathfrak{A} \models \alpha \Leftrightarrow \beta$ if it is the case that $\mathfrak{A} \models \alpha$ if and only if $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \exists x(\alpha)$ if there exists an element $d \in D_{\mathfrak{A}}$ such that $\mathfrak{A}[x : d] \models \alpha$;
- $\mathfrak{A} \models \forall x(\alpha)$ if for each element $d \in D_{\mathfrak{A}}$, it holds that $\mathfrak{A}[x : d] \models \alpha$;
- $\mathfrak{A} \models t = s$ if $t^{\mathfrak{A}} = s^{\mathfrak{A}}$.

For a given structure $\mathfrak{A}$, a symbol σ and a suitable value v for σ in $\mathfrak{A}$, we denote by $\mathfrak{A}[\sigma : v]$ the structure identical to $\mathfrak{A}$ except that $\sigma^{\mathfrak{A}[\sigma:v]} = v$.

Terminology (reminder)

If $\mathfrak{A} \models \varphi$, we also say that

- $\mathfrak{A}$ satisfies φ;
- $\mathfrak{A}$ is a model of φ;
- φ is true in $\mathfrak{A}$.

FO's theory of informal semantics

- An FO sentence φ over Σ contains logical and non-logical symbols.
- To determine the informal semantics of formulas over Σ, we first need to specify the meaning of the non-logical symbols.
- This is the intended interpretation of the symbols of Σ.
- The interpretation of the logical symbols is given:
  - In fact, their meaning $\neg, \wedge, \vee, \Rightarrow, \Leftrightarrow, \forall, \exists$ is explicitly specified in the definition of $\models$
  - The inductive definition of $\models$ is an recursive translation table.
- Thus, in principle, determining the empirical interpretation of FO sentences is easy.

Ex.

- The intended interpretation of the vocabulary:
  - Constant t: Tom;
  - Constant m: G0B23;
  - $P(x)$: x is a passing grades;
  - $G(x, y, z)$: student x has grade z for exam of course y.
- The sentence $\exists g(G(t, m, g) \wedge P(g))$ expresses:

Ex.

- The intended interpretation of the vocabulary:
  - Constant t: Tom;
  - Constant m: G0B23;
  - $P(x)$: x is a passing grades;
  - $G(x, y, z)$: student x has grade z for exam of course y.
- The sentence $\exists g(G(t, m, g) \wedge P(g))$ expresses:
  - There exists a g such that Tom has grade g for course G0B23 and g is a passing grade.

Ex.

- The intended interpretation of the vocabulary:
  - Constant t: Tom;
  - Constant m: G0B23;
  - $P(x)$: x is a passing grades;
  - $G(x, y, z)$: student x has grade z for exam of course y.
- The sentence $\exists g(G(t, m, g) \wedge P(g))$ expresses:
  - There exists a g such that Tom has grade g for course G0B23 and g is a passing grade.
  - I.e., Tom passes for course G0B23.

Ex.

- The intended interpretation of the vocabulary:
  - Constant t: Tom;
  - Constant m: G0B23;
  - $P(x)$: x is a passing grades;
  - $G(x, y, z)$: student x has grade z for exam of course y.
- The sentence $\exists g(G(t, m, g) \wedge P(g))$ expresses:
  - There exists a g such that Tom has grade g for course G0B23 and g is a passing grade.
  - I.e., Tom passes for course G0B23.

The informal semantics of a sentence under an intended interpretation is also called its declarative or intuitive reading.

The informal semantics

- FO as a kind of shorthand notation for a special subset of NL sentences.
- No poetry, but the sort of language found in mathematical texts.
    - precise!
- Impact for knowledge representation: One should be capable to reformulate information and knowledge in this style.
- This is sometimes surprisingly difficult. See secton below on pragmatics.

A first example: Group theory

A group is a mathematical structure consisting of a domain, a (total) binary operator and neutral element that satisfies three properties as expressed below.

- Vocabulary: $\Sigma = \{e/0 :, +/2 :\}$ – two function symbols with obvious intended interpretation.
- Axioms:
  - Associativity:
  $$\forall x \forall y \forall z (x + (y + z) = (x + y) + z)$$
  - Neutral element
  $$\forall x (x + e = x \wedge e + x = x)$$
  - Inverse element
  $$\forall x \exists y (x + y = e \wedge y + x = e)$$

A model of this theory represents a group.

To see the same theory written in IDP syntax:
`http://dtai.cs.kuleuven.be/krr/idp-ide/?present=group`
Push “run” to see all models $\mathfrak{A}$ with domain $\{a, b, c, d\}$ and $e^{\mathfrak{A}} = a$

Graph colouring

A graph consists of vertices and directed edges between them. Vertices are labeled by a color. Two adjacent vertices have different color.

- $\Sigma = \{Vertex/1, Edge/2, Col/1, Colour/1 :\}$ – obvious intended interpretations
- Theory

$$\begin{array}{l}\forall x \;\; Vertex(x) \Rightarrow Col(Colour(x)) \\ \forall x \forall y (Edge(x, y) \Rightarrow \neg(Colour(x) = Colour(y)))\end{array}$$

A model of this theory represents

Graph colouring

A graph consists of vertices and directed edges between them. Vertices are labeled by a color. Two adjacent vertices have different color.

- $\Sigma = \{Vertex/1, Edge/2, Col/1, Colour/1 :\}$ – obvious intended interpretations
- Theory

$$
\begin{array}{l}
\forall x \; Vertex(x) \Rightarrow Col(Colour(x)) \\
\forall x \forall y (Edge(x, y) \Rightarrow \neg(Colour(x) = Colour(y)))
\end{array}
$$

A model of this theory represents a set of colours and a correctly coloured graph.

Graph colouring

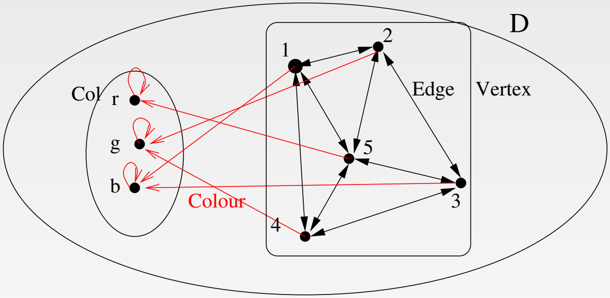
A graph consists of vertices and directed edges between them. Vertices are labeled by a color. Two adjacent vertices have different color.

- $\Sigma = \{Vertex/1, Edge/2, Col/1, Colour/1 :\}$ – obvious intended interpretations
- Theory

$$\begin{array}{l} \forall x\ Vertex(x) \Rightarrow Col(Colour(x)) \\ \forall x \forall y (Edge(x,y) \Rightarrow \neg(Colour(x) = Colour(y))) \end{array}$$

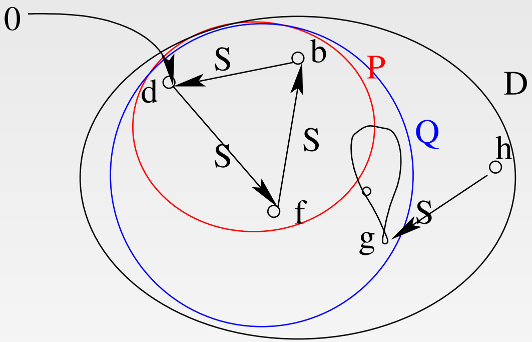
A model of this theory represents a set of colours and a correctly coloured graph.
But how do we solve a specific graph colouring problem using it?
The answer follows ..

A model in graphical notation



Note:

- values for every symbol
- the value of *Colour* maps each color to itself. Indeed, since it should be a total function, the maps of each color should exist. However, the value is an arbitrary choice.



Homework 1: What does the following proposition mean:

$$P(0) \wedge \forall x (P(x) \Rightarrow P(S(x)))$$

Verify whether this structure satisfies it. Propose an alternative interpretation of P such that the formula is not satisfied. If we replace P by Q, is the formula true? Express that P is a subset of Q as a formula. Try to express that g cannot be reached from 0 through the function S. Does it work?

Abstract syntax

- Syntax is not very important.
- Other syntaxes, same abstract syntax:

  FORALL x: IF x $\in$ human THEN x $\in$ woman OR x $\in$ man

  $$\forall x(\mathit{human}(x) \Rightarrow \mathit{woman}(x) \vee \mathit{man}(x))$$

  FORALL x: x $\in$ adult IFF x $\in$ human AND x.age $>$ 18

  $$\forall x(\mathit{adult}(x) \Leftrightarrow \mathit{human}(x) \wedge \mathit{age}(x) > 18)$$

  Which do you prefer? Would it make a big difference for you?
  It is no big deal in principle.

Some fundamental properties

- FO's semantics satisfies Frege's compositionality principle
  - the meaning of a connective does not depends on the context in which it occurs.
- Connectives of FO are truth functional:
  - The value of a composite formula is computed by applying a boolean function on the value of the component formulas.

E.g., a non-truth functional natural language connective:
"I went home <u>and then</u> I called her":
Homework 2: Do a possible world analysis. Think up two worlds in which the two subsentences are true but the value of the entire sentence is different.

Is the informal semantics of FO really precise?

Overloading and ambiguity

Like other words in natural language, many natural language connectives have multiple meanings. They are overloaded.

- E.g., the word "and" has a different meaning in the following two sentences.
  - The earth is a planet and the sun a star.
  - I got up this morning and brushed my teeth.

  The first and is the standard "logical" conjunction. The second and is the temporal conjunction that states that the first conjunct precedes the second in time.
- Similarly, "or" in natural language may be intended inclusively or exclusively.
  - As a reward, you can choose a bike or a train subscription. (one only)
  - Hillary or Bill will be home. (At least one, perhaps both).

- Notice that these statements were not ambiguous. You figgered out what was meant and correctly determined the intended meaning of the connective. (Or so I hope for you).
- Our subconscious language interpreting brain module is extremely strong in disambiguating overloaded words.
- Thus, overloading does not necessarily lead to ambiguity. Sometimes it does, but often it does not. In carefully phrased text, as we find it in science and mathematics, or in law texts, we avoid ambiguity even if we use overloaded words, by providing enough context to disambiguate them.

- In logic, every connective has a unique meaning. It is not overloaded, and that is how it should be in a formal language.
  - If a different meaning of some natural language connective is needed, a new connective should be added (e.g., exclusive disjunction).
- In the definition of satisfaction, we define the meaning of formal connectives in terms of disambiguated versions of the natural language connective. E.g., the inductive rule:

  $$\mathfrak{A} \models \varphi \wedge \psi \text{ if } \mathfrak{A} \models \varphi \text{ and } \mathfrak{A} \models \psi$$

  Here the logical and is meant, not the temporal one. E.g., the inductive rule:

  $$\mathfrak{A} \models \varphi \vee \psi \text{ if } \mathfrak{A} \models \varphi \text{ or } \mathfrak{A} \models \psi \text{ (or both)}$$

  We had added "or both" to disambiguate the disjunction as inclusive.
- In case any ambiguity is left (which I doubt), the truth tables for connectives and quantifiers eliminates it.

The conditional and material implication

- One of the most overloaded connectives is “if . . . then . . . ”.
- This is called the conditional in linguistics.
- Linguistics have distinguished more than 30 different meanings and uses.
- Material implication, as embodied in FO, captures just one of these meanings!
- Unavoidably, some conditional statements in natural language will not be correctly represented by material implication in FO.
- Sometimes, this may be very subtle, as we shall see.
- We will later introduce another conditional in this chapter.
- The many meanings of the conditional have caused quite a bit of confusion and controversy about material implication.
- Yet, the material implication covers most applications of “if. . . then..”, certainly those seen in this course. It is probably the most common form of conditional. We really need it!

Intermezzo: material implication

- Compare:
  - If you succeed for all courses, then you pass.
    - Proposed formalization:

$$(\forall c\ \mathit{Succ}(c)) \Rightarrow \mathit{Pass}$$

  - There is a course such that, if you succeed for it, you pass
    - Proposed formalization:

$$\exists c(\mathit{Succ}(c) \Rightarrow \mathit{Pass})$$

- Are the NL statements equivalent?

Intermezzo: material implication

- Compare:
  - If you succeed for all courses, then you pass.
    - Proposed formalization:

$$(\forall c\ Succ(c)) \Rightarrow Pass$$

  - There is a course such that, if you succeed for it, you pass
    - Proposed formalization:

$$\exists c(Succ(c) \Rightarrow Pass)$$

- Are the NL statements equivalent? No!

Intermezzo: material implication

- Compare:
  - If you succeed for all courses, then you pass.
    - Proposed formalization:

$$(\forall c\ \mathit{Succ}(c)) \Rightarrow \mathit{Pass}$$

  - There is a course such that, if you succeed for it, you pass
    - Proposed formalization:

$$\exists c(\mathit{Succ}(c) \Rightarrow \mathit{Pass})$$

- Are the NL statements equivalent? No!

  By possible world analysis: our university satisfies the first but not the second formula.
- Are the proposed formalizations logically equivalent?

Intermezzo: material implication

- Compare:
  - If you succeed for all courses, then you pass.
    - Proposed formalization:

$$(\forall c \ Succ(c)) \Rightarrow Pass$$

  - There is a course such that, if you succeed for it, you pass
    - Proposed formalization:

$$\exists c(Succ(c) \Rightarrow Pass)$$

- Are the NL statements equivalent? No!

  By possible world analysis: our university satisfies the first but not the second formula.
- Are the proposed formalizations logically equivalent? Yes! :
  $\exists c(Succ(c) \Rightarrow Pass) \equiv (\forall c \ Succ(c)) \Rightarrow Pass$.

Intermezzo: material implication

- Compare:
  - If you succeed for all courses, then you pass.
    - Proposed formalization:

$$(\forall c\ \mathit{Succ}(c)) \Rightarrow \mathit{Pass}$$

  - There is a course such that, if you succeed for it, you pass
    - Proposed formalization:

$$\exists c(\mathit{Succ}(c) \Rightarrow \mathit{Pass})$$

- Are the NL statements equivalent? No!

  By possible world analysis: our university satisfies the first but not the second formula.
- Are the proposed formalizations logically equivalent? Yes! : $\exists c(\mathit{Succ}(c) \Rightarrow \mathit{Pass}) \equiv (\forall c\ \mathit{Succ}(c)) \Rightarrow \mathit{Pass}$.
- This means that at least one of these FO sentences does not correctly formalize the corresponding NL-statement. Indeed, the problem is with $\exists c(\mathit{Succ}(c) \Rightarrow \mathit{Pass})$.

Intermezzo: material implication

The following derivation is a correct proof:

$\exists c(Succ(c) \Rightarrow Pass)$
is logically equivalent to $\exists c(\neg Succ(c) \vee Pass)$
is logically equivalent to $(\exists c(\neg Succ(c))) \vee Pass)$
is logically equivalent to $(\neg \forall c\ Succ(c)) \vee Pass)$
is logically equivalent to $(\forall c\ Succ(c)) \Rightarrow Pass$

Every step is correct. The only reason why it surprises us is that one of the original sentences does not mean what we think it means.

Intermezzo: material implication

Homework 3: What is wrong with the formula $\exists c(\mathit{Succ}(c) \Rightarrow \mathit{Pass})$, intuitively?

The paradox on the previous slides is similar to Smullyans drinkers paradox. `https://en.wikipedia.org/wiki/Drinker_paradox`

Equivalences

- The equivalence sign is misleadingly simple. It is easy to make mistakes that are hard to trace.
- E.g., we define a person has fever if his temperature is more than 37.

$$\forall x \forall t (Fever(x) \Leftrightarrow Temperature(x, t) \wedge t > 37)$$

Correct?

Equivalences

- The equivalence sign is misleadingly simple. It is easy to make mistakes that are hard to trace.
- E.g., we define a person has fever if his temperature is more than 37.

$$\forall x \forall t (\mathit{Fever}(x) \Leftrightarrow \mathit{Temperature}(x, t) \wedge t > 37)$$

Correct? Not even close.

- Let us split in equivalent statement:

$$\forall x \forall t (Fever(x) \Leftarrow Temperature(x,t) \wedge t > 37) \wedge$$
$$\forall x \forall t (Fever(x) \Rightarrow Temperature(x,t) \wedge t > 37)$$

  Because t does not occur in the premisse of sentence 2, it is logically equivalent to:

$$\forall x (Fever(x) \Rightarrow \forall t (Temperature(x,t) \wedge t > 37)$$

  This says that if some x has fever, every object t in the universe is the temperature of x and also, that t is also larger than 37. Incorrect!

- How to correct?

$$\forall x (Fever(x) \Leftrightarrow \exists t (Temperature(x,t) \wedge t > 37))$$

Assume that we want to define *Even* to be exactly the set of all even numbers, that is two-folds of natural numbers *Nat*. Here is a proposed formula:

$$\forall n(Even(2 \times n) \Leftrightarrow Nat(n))$$

Is this a correct representation?

Assume that we want to define *Even* to be exactly the set of all even numbers, that is two-folds of natural numbers *Nat*. Here is a proposed formula:

$$\forall n(Even(2 \times n) \Leftrightarrow Nat(n))$$

Is this a correct representation? Let us verify.

- $\forall n(Even(2 \times n) \Rightarrow Nat(n))$: this is true if *Even* is the set of even numbers. Notice that it is also true if *Even* is a superset of the even numbers.
- $\forall n(Even(2 \times n) \Leftarrow Nat(n))$: this is true if *Even* is the set of even numbers. But it is also true if *Even* is a superset of the even numbers.

E.g., take the structure $\mathfrak{A}$, where $Nat^{\mathfrak{A}} = Even^{\mathfrak{A}} = \mathbb{N}$, the set of natural numbers, and $2^{\mathfrak{A}}, \times^{\mathfrak{A}}$ have the expected value. Clearly, this structure does not satisfy the definition, and yet, it satisfies the equivalence.

We intend to define narcistic love as self love. That is: $NarcisticLove = \{(n,n) | Love(n,n)\}$: follows:

$$\forall x (NarcisticLove(x,x) \Leftrightarrow Love(x,x))$$

We intend to define narcistic love as self love. That is: $NarcisticLove = \{(n, n) | Love(n, n)\}$: follows:

$$\forall x (NarcisticLove(x, x) \Leftrightarrow Love(x, x))$$

Completely wrong! See: `http://dtai.cs.kuleuven.be/krr/idp-ide/?present=BadEquiLove`

We intend to define narcistic love as self love. That is: $NarcisticLove = \{(n,n) | Love(n,n)\}$: follows:

$$\forall x (NarcisticLove(x,x) \Leftrightarrow Love(x,x))$$

Completely wrong! See: `http://dtai.cs.kuleuven.be/krr/idp-ide/?present=BadEquiLove`

We define that P is the set of all sums $n+m$ with $n,m \in Q$:

$$\forall n \forall m (P(n+m) \Leftrightarrow Q(n) \wedge Q(m))$$

We intend to define narcistic love as self love. That is: $NarcisticLove = \{(n, n) | Love(n, n)\}$: follows:

$$\forall x(NarcisticLove(x, x) \Leftrightarrow Love(x, x))$$

Completely wrong! See: `http://dtai.cs.kuleuven.be/krr/idp-ide/?present=BadEquiLove`

We define that P is the set of all sums $n + m$ with $n, m \in Q$:

$$\forall n \forall m(P(n + m) \Leftrightarrow Q(n) \wedge Q(m))$$

Completely wrong! See: `http://dtai.cs.kuleuven.be/krr/idp-ide/?present=BadEquiSum`

The openness of FO

- Unlike many other logics, FO makes no hidden assumptions.
  - FO has the open domain assumption: the domain is arbitrary: maybe finite or infinite. Unless an axiom states differently.
  - FO has the free term assumption: any pair of terms could be equal or not. Unless an axiom states differently.
  - FO makes no default assumption about the truth of atoms, or any connection between them. They can be true or false, in arbitrary combination. Unless an axiom states differently.
- This openess is visible in its formal semantics (see next slide).
- If information is known about domain, identity of terms, truth or falsity of atoms, then this information should be explicitly expressed in FO
- This openess has its advantages and its disadvantages.

- Where does FO's openess come from, semantically?
  - From the unconstrained nature of the class of structures that FO admits.
- The class of models of the empty theory over vocabulary Σ is the class of all Σ-structures. This is big!
  - Structures with arbitrary domain, of arbitrary size (finite, countably infinite, uncountably infinite), with arbitray interpretation of function symbols, and predicate symbols. They abstractly represent more states of affairs than we sometimes can imagine.
- For any such structure $\mathfrak{A}$, if we do not consider it to be an abstraction of a possible state of affairs, we should add an axiom to our theory that is violated in $\mathfrak{A}$. If we do not, then FO "thinks" this state of affairs is possible.

This causes FO not to make no assumptions at all.

In other modelling logics, the class of models is typically much more constrained.

- The structures of logic programs are Herbrand structures (their domain is the set of all terms over Σ, and the value of different terms is always different).

As said, this has its advantages and disadvantages. (See below).

Use case: expressing information from a database in FO

Consider two tables from a database:

Instructor/2	
Ray	CS230
Hec	CS230
Wal	HD87
Mar	HD87

Enrolled/2	
Jill	CS230
Jack	CS230
Tom	HD87

Can we express this information in FO?

Use case: expressing information from a database in FO

- First idea:
  - Define Σ consisting of predicate symbols $\mathit{Instructor}/2, \mathit{Enrolled}/2$ and constant symbols $\mathit{Ray}, \mathit{Hec}, \ldots, \mathit{CS}230, \ldots, \mathit{Jill}, \ldots$.
  - Express information by the following conjunction that enumerates all table rows as a conjunction of atoms:

$$\mathit{Instructor}(\mathit{Ray}, \mathit{CS}230) \wedge \cdots \wedge \mathit{Enrolled}(\mathit{Jill}, \mathit{CS}230) \wedge \ldots (*)$$

Use case: expressing information from a database in FO

- First idea:
  - Define Σ consisting of predicate symbols $\mathit{Instructor}/2, \mathit{Enrolled}/2$ and constant symbols $\mathit{Ray}, \mathit{Hec}, \ldots, \mathit{CS}230, \ldots, \mathit{Jill}, \ldots$.
  - Express information by the following conjunction that enumerates all table rows as a conjunction of atoms:

$$\mathit{Instructor}(\mathit{Ray}, \mathit{CS}230) \wedge \cdots \wedge \mathit{Enrolled}(\mathit{Jill}, \mathit{CS}230) \wedge \ldots (*)$$

- This is not correct:
  - The database contains negative information, e.g.,
    - that different constant symbols $\mathit{Ray}, \mathit{Hec}, \mathit{CS}230, \ldots$ represent different objects;
    - that facts not in table are false, e.g, $\mathit{Instructor}(\mathit{HD}87, \mathit{HD}87)$ is false
  - In contrast, the above formula contains no negative information that was available in the database.

A correct modelling

Two sorts of axiomas are needed

- Equality axioms to express different constants denote different objects: UNA
- Precise definitions of *Instructor* and *Enrolled*, that describe what is in and what is out the tables.

UNA axiom

- UNA: Unique Names Axioms:
- Let Σ be a set of constant symbols. Then $UNA(\Sigma)$ is the conjunction of axioms:

$$\neg C_1 = C_2$$

  where $C_1, C_2 \in \Sigma, C_1 \neq C_2$.
- Note that this is a quadratic number of axioms in the size of Σ. Obviously, FO should be extended with a new construct to express this axiom compactly.
- In IDP, this is done by constructed types (see below).

Predicate completion

For the predicates, we need an axiom that exactly expresses which tuples are in it and which are out:

$$\begin{aligned}\forall x(\forall y(&\mathit{Instructor}(x,y) \Leftrightarrow\\ &(x = \mathit{Ray} \wedge y = \mathit{CS}230)\vee\\ &(x = \mathit{Hec} \wedge y = \mathit{CS}230)\vee\\ &(x = \mathit{Wal} \wedge y = \mathit{HD}87)\vee\\ &(x = \mathit{Mar} \wedge y = \mathit{HD}87)))\end{aligned}$$

We call this a completed definition of *Instructor*/2.
A similar completed definition is needed for *Enrolled*.

The combination of UNA and the two completed definitons can be shown to capture all the information in the two tables.

Evaluation:

- in many modelling applications, a possibly large number of objects need to be modelled. Symbols need to be chosen as identifiers for objects. In pure FO, we need to explicitly add UNA axioms for all these constants.
- In other languages such as logic programming, Answer Set Programming, UNA is implicitly imposed. In this case, constants and function symbols become constructors
- we cannot use them anymore to express real functions or constants of free identity.
  - E.g., constants $Obama, PresidentOfUs$ in the obvious intended interpretation in the actual world denote the same individual. Thus, UNA cannot be imposed on these atoms.

How to avoid UNA in IDP?

- Constructors and constructed types:

  ```
  type day constructed from { mon; tue; wed; thu;
                              fri; sat; sun }
  ```

  this logically means:

  - $\forall x(day(x) \Leftrightarrow x = mon \vee \cdots \vee x = sun)$
  - $UNA(\{mon, \ldots, sun\})$

How to avoid UNA in IDP?

- Constructors and constructed types:

```
type day constructed from { mon; tue; wed; thu;
                fri; sat; sun }
```

  this logically means:
  - $\forall x(day(x) \Leftrightarrow x = mon \vee \cdots \vee x = sun)$
  - $UNA(\{mon, \ldots, sun\})$

- The object identifiers in structures are assumed to denote different objects. They are not symbols of the vocabulary and cannot be used in the theory.

In the future, we want more flexible schemes to combine the advantages of identifiers and constructors (implicit UNA) with non-constructor constant and function symbols.

- As mentioned before, we believe that FO's connectives and quantifiers are fundamental for KR.
- However, it does not suffice for knowledge representation.

Typed FO: FO(Types)

- In standard FO, there is a unique base type where all quantification, functions and predicates range over.
- Just like in other languages, types in FO can often be useful
  - (1) to improve the precision of the modeling and
  - (2) reduce the amount of human errors.

  E.g., in the graph colouring, there are two natural types: vertices and colors. We would like to define the colouring function as a function from vertices to colors so that it is not defined on colors.
- It is straightforward to extend FO with types.

FO(Types)

- The IDP system supports typed logic with subsorts.
- IDP performs type derivation for variables. That is, it "guesses" the type of variables from the positions in which it occurs.

Example

Graph-colouring:

- Types `Vertex` and `Col`
- Symbols `G(Vertex,Vertex), Colour(Vertex):Col`
- A well-typed axiom
  `!  x y :  (G(x,y) => Colour(x)~=Colour(y))`
  Variables `x, y` are derived to be of type `Vertex`.
- Explicit typing:
  `!  x[Vertex] y[Vertex] :  (G(x,y) => Colour(x)~=Colour(y))`
- Badly typed on variable `y`:
  `!  x y :  (G(x,y) => Colour(x)=y)`

FO(Types,Arit)

- FO(Types) is a basis for adding interpreted types.
- FO(Types,Arit) is obtained by adding the integer numbers:
  - type symbol *int*
  - numerals: strings $0, 12, -5, \ldots$
  - arithmetic operator symbols $+, \times, \hat{}\,(\text{power}), <, \leq, >, \geq, \ldots$
- Their value in every structure $\mathfrak{A}$ is the obvious one.
  - E.g., the value $12^{\mathfrak{A}}$ of numeral 12 in $\mathfrak{A}$ is the number 12.
  - E.g., the value $<^{\mathfrak{A}}$ of symbol $<$ in $\mathfrak{A}$ is the standard strict order relation $<$ on integers, that is, $\{(n, m) \in \mathbb{Z}^2 \mid n < m\}$.

Notice the difference between a symbol and a value. Here, we used colors to distinguish between them , e.g., symbols 12, $\leq$ and values 12 and $\leq$.

FO(Types,Arit)

- Now we can use integer arithmetic in our theories.
  - E.g., Fermats last theorem is now formalized as follows:

$$\neg\exists x\exists y\exists z\exists n(n > 2 \land x\,\hat{}\,n + y\,\hat{}\,n = z\,\hat{}\,n)$$

  (It is true, but was not easy to prove ($>$300years).)
- We can do the same for other types of numbers $\mathbb{N}, \mathbb{Q}, \mathbb{R}$.
- IDP currently only supports finitely bounded arithmetic. Every numerical symbol has to be declared to be an element of a finite subtype of integers.

IDP example

```
vocabulary V{
   type someIntegers isa int
   C1 :  someIntegers
   C2 :  someIntegers
}
theory T: V{
   C1 + C1 = C2.
}
structure S:V{
   someIntegers = {1..5}
   }
```

- a subtype declaration using `isa`
- A bounded integer type declaration `{1..5}`

FO(Agg)

- Sometimes, it is useful to be able to talk about properties of sets, e.g., its cardinality.

  $$\forall c \forall r (Course(c) \land Room(c, r) \Rightarrow \#\{y : Enr(y, c)\} \leq Capac(r))$$

  The room in which some course is taught must be larger than the number of students enrolled in the course.
- $\#$ is the cardinality aggregate and represents the function from sets to natural numbers.
- It takes as argument a set expression $\{(x_1, \ldots, x_n) : \varphi\}$.
- It denotes the cardinality of the set represented by this set expression.

FO(Agg)

Useful aggregates:

- $\#\{(x_1, \ldots, x_n) : \varphi\}$ represents the function mapping a set to its number of elements.
- $Min\{x : \varphi\}$ represents the function mapping a set of integer numbers to its least number.
- $Max\{x : \varphi\}$ represents the function mapping a set of natural numbers to its largest number.
- $Sum\{(x, x_1, \ldots, x_n) : \varphi\}$ represents the function from a set of tuples to the sum of the first arguments of these tuples:

$$Sum\{(x, x_1, \ldots, x_n) : \varphi\} = \sum_{\{(x, x_1, \ldots, x_n) : \varphi\}} x$$

The sum aggregate

- Expressing that the study load of a student should be less than 65 study points:

  $$\forall s(Sum\{(l,c) : SelCourse(s,c) \land StudyPoints(c,l)\} \leq 65)$$

  In words: the sum of the first arguments of all tuples in the set of tuples (l,c) such that l is the number of study points of a course followed by a student s should be less than 65, for each student s.
- Defining the total studyload $StudyL/1:$ of a student:

  $$\forall s\ StudyL(s) = Sum\left\{(l,c) : \left(\begin{array}{l} SelCourse(s,c)\land \\ StudyPoints(c,l) \end{array}\right)\right\}$$

The sum aggregate

$$\forall s (Sum\{(l, c) : SelCourse(s, c) \land StudyPoints(c, l)\} \leq 65)$$

- Why do we sum over the first arguments of tuples, why do we need the other arguments?

The sum aggregate

$$\forall s(Sum\{(l,c) : SelCourse(s,c) \wedge StudyPoints(c,l)\} \leq 65)$$

- Why do we sum over the first arguments of tuples, why do we need the other arguments?
    - Compare the above constraints with the following one:

      $$\forall s(Sum\{l : \exists c(SelCourse(s,c) \wedge StudyPoints(c,l))\} \leq 65)$$

The sum aggregate

$$\forall s(Sum\{(l,c) : SelCourse(s,c) \land StudyPoints(c,l)\} \leq 65)$$

- Why do we sum over the first arguments of tuples, why do we need the other arguments?
  - Compare the above constraints with the following one:
    $$\forall s(Sum\{l : \exists c(SelCourse(s,c) \land StudyPoints(c,l))\} \leq 65)$$
  - Imagine that a student *Bob* follows 10 courses c1,..,c10 of 8 study points each. His total load is 80, far above the threshold.

The sum aggregate

$$\forall s(Sum\{(l,c) : SelCourse(s,c) \wedge StudyPoints(c,l)\} \leq 65)$$

- Why do we sum over the first arguments of tuples, why do we need the other arguments?
  - Compare the above constraints with the following one:
    $$\forall s(Sum\{l : \exists c(SelCourse(s,c) \wedge StudyPoints(c,l))\} \leq 65)$$
  - Imagine that a student *Bob* follows 10 courses c1,..,c10 of 8 study points each. His total load is 80, far above the threshold.
  - Now compare the sets
    $\{(l,c) : SelCourse(Bob,c) \wedge StudyPoints(c,l)\}$
    $\{l : \exists c(SelCourse(Bob,c) \wedge StudyPoints(c,l))\}$
    What is the sum of these sets?

The sum aggregate

$$\forall s(Sum\{(l,c) : SelCourse(s,c) \wedge StudyPoints(c,l)\} \leq 65)$$

- Why do we sum over the first arguments of tuples, why do we need the other arguments?
  - Compare the above constraints with the following one:
    $$\forall s(Sum\{l : \exists c(SelCourse(s,c) \wedge StudyPoints(c,l))\} \leq 65)$$
  - Imagine that a student *Bob* follows 10 courses c1,..,c10 of 8 study points each. His total load is 80, far above the threshold.
  - Now compare the sets
    $\{(l,c) : SelCourse(Bob,c) \wedge StudyPoints(c,l)\}$
    $\{l : \exists c(SelCourse(Bob,c) \wedge StudyPoints(c,l))\}$
    What is the sum of these sets?
  - Respectively 80 and 8. Therefore, the second constraint is wrong, is too weak. By adding more arguments, we can distinguish between different occurrences of the same study load for different courses.

Extending FO to FO(Agg)

Syntax:

- If $\mathbf{x}$ is a tuple of variable symbols, φ a FO-formula, then $\{\mathbf{x} : \varphi\}$ is a set expression
- If s is a set expression and Agg is an aggregate symbol (e.g., $\#, Sum, Min, \ldots$) then $Agg(s)$ is a term (called an aggregate term)

Semantics:

- $\{\mathbf{x} : \varphi\}^{\mathfrak{A}} = \{\bar{d} | \mathfrak{A}[\mathbf{x} : \bar{d}] \models \varphi\}$
- $Agg(s)^{\mathfrak{A}} = Agg^{\mathfrak{A}}(s^{\mathfrak{A}})$

(That is all it takes in FO.)

In IDP, aggregates are represented in a slightly different syntax:

- number of elements of P

```
    #{x,y:  P(x,y)}.
```

In IDP, aggregates are represented in a slightly different syntax:

- number of elements of P

  ```
  #{x,y:  P(x,y)}.
  ```

- sum of $x+y$, for all $(x,y) \in P$

  ```
  sum{x,y:  P(x,y) :  x+y}.
  ```

In IDP, aggregates are represented in a slightly different syntax:

- number of elements of P
  ```
  #{x,y:  P(x,y)}.
  ```
- sum of x+y, for all (x,y)∈ P
  ```
  sum{x,y:  P(x,y) :  x+y}.
  ```
- minimum of set { x : Q(x)& R(x)}
  ```
  min{x:  Q(x)& R(x) :  x}.
  ```

In IDP, aggregates are represented in a slightly different syntax:

- number of elements of P
  ```
  #{x,y:  P(x,y)}.
  ```
- sum of x+y, for all (x,y)∈ P
  ```
  sum{x,y:  P(x,y) :  x+y}.
  ```
- minimum of set { x : Q(x)& R(x)}
  ```
  min{x:  Q(x)& R(x) :  x}.
  ```
- maximum :
  ```
  max{x:  Q(x)& R(x) :  x}.
  ```

In IDP, aggregates are represented in a slightly different syntax:

- number of elements of P
  ```
  #{x,y:  P(x,y)}.
  ```
- sum of x+y, for all (x,y)∈ P
  ```
  sum{x,y:  P(x,y) :  x+y}.
  ```
- minimum of set { x : Q(x)& R(x)}
  ```
  min{x:  Q(x)& R(x) :  x}.
  ```
- maximum :
  ```
  max{x:  Q(x)& R(x) :  x}.
  ```
- Nesting is allowed, as in:
  ```
  Pnest= sum{x[num]:x=#{y:Q(x,y)} :  x }.
  ```

`http://dtai.cs.kuleuven.be/krr/idp-ide/?present=Agg`

Homework 4: Experiment with different input/output.

- Compute value aggregate expressions from values for P, R.
- Compute values of P, R from value aggregates expressions.
- Compute minimal structure satisfying aggregates expressions.

There is a difference in the theoretical language en the concrete language of IDP.

Definitions in KR

- Together with axioms (equations), definitions are a basic component of scientific and mathematical theories.
  E.g., an example of a definition in the gravitation theory in Chapter II is that of the unity vectors $\bar{V}_{ij}$ which were defined (by an equation) as the vector of length 1 in the direction from body i to j.
- Frequently, we need auxiliary concepts that can be defined in terms of more basic ones. We introduce a new symbol and define it in terms of the existing one.
- Some definitions are inductive/recursive.
- It is well-known that inductive definitions in general cannot be expressed in FO (and we will prove it later in this course). Therefore, the last extension of FO that we consider here, is a language construct for definitions.

Proof idea:

- Take a proposition that can be expressed using inductive definitions

  There is a path from vertex A to vertex B in graph G

- Show that the proposition cannot be expressed in FO.

$\mathcal{M}od(T)$: the class of models of theory T.

Definition

A theory T over Σ expresses a class of Σ-structures $\mathcal{C}$ if $\mathcal{M}od(T) = \mathcal{C}$.

The compactness theorem of FO

Compactness Theorem of FO

An infinite FO theory Ψ is unsatisfiable iff at least one finite subset is unsatisfiable.

- The compactness theorem is historically the first technique to prove inexpressibility in FO.
- More advanced techniques exist. E.g., Ehrenfeucht-Fraïssé Games.

Unreachability can be infinitely expressed

"There is no path from A to B in graph G"

- $\Sigma = \{A, B, G/2\}$
- The proposition is expressed by the theory $T_{UnR} =$

$$\left\{ \begin{array}{l} \neg G(A,B), \\ \neg \exists x_1 (G(A,x_1) \wedge G(x_2,B)), \\ \dots \\ \neg \exists x_1 \dots \exists x_n (G(A,x_1) \wedge \dots \wedge G(x_n,B)), \\ \dots \end{array} \right\}$$

Proof. The propositions express: there is no path of length 1, of length 2, of length 3, ... A structure in which there is no path from A to B satisfies each of these propositions, and vice versa, a structure that satisfies each of these propositions cannot have a path from A to B.

Reachability cannot be expressed

Theorem

"There is a path from A to B in graph G" is not expressible in FO.

Proof. Let $\Sigma = \{A, B, G/2\}$. Every Σ-structure specifies a graph and vertices corresponding to A, B.

- Consider the class $\mathcal{C}_R$ of Σ-structures $\mathfrak{A}$ with a path from $A^{\mathfrak{A}}$ to $B^{\mathfrak{A}}$ in graph $G^{\mathfrak{A}}$.
- Assume $\mathcal{C}_R$ is expressible by FO theory T_R. Then the infinite theory $T_R \cup T_{UnR}$ is unsatisfiable.
- Hence, it has an unsatisfiable finite subset Ω.
- Let n be the largest path length forbidden by Ω. Every structure with a shortest path from A to B that is strictly longer than n satisfies Ω.
- Contradiction

- Reachability, transitive closure are important concepts in many applications, but they cannot be expressed in FO.
- Adding inductive definitions to FO will solve this problem.

It will be expressed by the FO(ID) theory:

$$\left\{ \begin{array}{l} R(A). \\ \forall x (R(x) \leftarrow \exists y (R(y) \land G(y,x))) \end{array} \right\}$$
$$R(B)$$

A language construct for expressing (inductive) definitions

- To be able to express also inductive definitions, we introduce a new language construct, inspired by the way inductive definitions are presented in mathematical text.
- Two prototypical examples:

The reachability relation R of G is defined inductively:
- $(x, y) \in R$ if $(x, y) \in G$;
- $(x, y) \in R$ if for some vertex z,

$$(x, z), (z, y) \in R.$$

We define $\mathfrak{A} \models \varphi$ by induction:
- $\mathfrak{A} \models P(t_1, \ldots, t_n)$ if $(t_1^{\mathfrak{A}}, \ldots, t_n^{\mathfrak{A}}) \in P^{\mathfrak{A}}$
- ...
- $\mathfrak{A} \models \alpha \wedge \beta$ if $\mathfrak{A} \models \alpha$ and $\mathfrak{A} \models \beta$;
- $\mathfrak{A} \models \neg\alpha$ if $\mathfrak{A} \not\models \alpha$ - ...

- They are often presented as a collection of rules, providing base cases and inductive cases.