Early Knowledge Representation Formalisms [1]

Uli Sattler

General Remarks

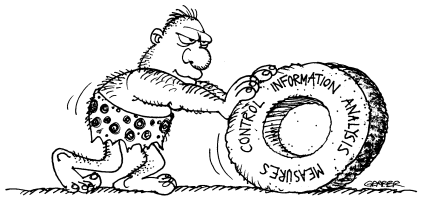
The following is very important:

- if you have a question, please ask me!
- if you have difficulties understanding me/reading my writing/... let me know!
- if this course is too slow/boring: let me know!
- if this course is too fast/difficult: let me know!
- to follow this course successfully, you might have to **think** a lot:
  we will talk about complex stuff such as representation of and reasoning about knowledge — and there are many facets to that and many ways to do it!
- Intended Learning Outcomes (short version):
  - you have seen and experienced different formalisms for knowledge representation and reasoning
  - you are aware of the different aspects/issues in KR
  - you can go and find an appropriate KR formalism if you need one—and you will not try to build one on your own if avoidable!

General Remarks

In a nutshell, what I try to show you is that

- a lot of people have already thought a lot about how to represent knowledge and how to reason about it and
- have developed quite sophisticated, powerful formalisms –
- doing it yourself can be fun, exciting, and provide insight,
- but I also want to prevent you from re-inventing the wheel



General Remarks

In a nutshell, what I try to show you is that

- a lot of people have already thought a lot about how to represent knowledge and how to reason about it and
- have developed quite sophisticated, powerful formalisms –
- doing it yourself can be fun, exciting, and provide insight,
- but I also want to prevent you from re-inventing the wheel:
  be aware of and use the toolbox that is already out there!

General Remarks

In a nutshell, what I try to show you is that

- a lot of people have already thought a lot about how to represent knowledge and how to reason about it and
- have developed quite sophisticated, powerful formalisms –
- doing it yourself can be fun, exciting, and provide insight,
- but I also want to prevent you from re-inventing the wheel:
  be aware of and use the toolbox that is already out there!

What is Knowledge Representation (KR)?

One answer: KR is the study of

- how **knowledge** about the world can be represented and
- what kinds of **reasoning** can be done with that knowledge.

Another answer: KR is the branch of **artificial intelligence** (AI) that deals with the construction, description and use of **ontologies**

So: What exactly is

- **knowledge**, i.e., what is the difference between
  - data,
  - information, and
  - knowledge?
- **artificial intelligence**?
- **ontologies**?

What is Knowledge Representation (KR)?

Many different answers, e.g.:

- **Data:**
  - the term **data** is often used to refer to the information stored in the computer
  - facts represented in a readable language (such as numbers, characters, images, or other methods of recording) on a durable medium. Data on its own carries no meaning.
- **Information:**
  - data that has been **interpreted**, translated, or transformed to reveal the underlying meaning
  - a message received and understood
  - a collection of facts from which conclusions may be drawn

What is Knowledge Representation (KR)?

Many different answers, e.g.:

- **Knowledge:**
  - the psychological result of perception and learning and reasoning
  - the body of truth, information, and principles acquired by humans
  - interpreted information that can be used
  - information evaluated and organised in the human mind so that it can be used purposefully
- **Artificial Intelligence:**
  - the use of computer algorithms, models, and systems to emulate human perception, cognition, and reasoning
  - a property of machines that, if achieved, mimics human thought processes. Many researchers in AI consider the abilities of "learning", reasoning, and decision making as essential to claims of machines possessing intelligence

What is Knowledge Representation (KR)?

Many different answers, e.g.:

- **Artificial Intelligence:** (ctd.)
  - software that is 'intelligent', or that lends 'intelligence' to a machine or computer. 'Intelligence' might be deemed to be the computational part of the ability to solve problems and achieve goals.
- **Knowledge Representation:**
  - the study of how knowledge about the world can be represented and what kinds of reasoning can be done with that knowledge.
  - the branch of AI that deals with the construction, description and use of ontologies.
  - the notation or formalism used for coding the knowledge to be stored in a knowledge-based system.
  - a structure in which knowledge can be stored that allows the system to understand the relationships among pieces of knowledge and to manipulate those relationships

What is Knowledge Representation (KR)?

Many different answers, e.g.:

- **Ontologies:**
  - an explicit formal specification of how to represent the objects, concepts and other entities that are assumed to exist in some area of interest and the relationships that hold among them
  - a formal, explicit specification of a shared conceptualization.
    - "Conceptualization" refers to an abstract model of phenomena in the world by having identified the relevant concepts of those phenomena.
    - "Explicit" means that the type of concepts used, and the constraints on their use are explicitly defined.
    - "Formal" refers to the fact that the ontology should be machine readable.
    - "Shared" reflects that ontology should capture consensual knowledge accepted by the communities.

Basic Ingredients of KR

- in KR, we are concerned with **formalisms** to represent knowledge and reason about knowledge possibly in/with a computer
- KR aims at building/investigating powerful, useful formalisms to
  - "teach" computers how to think instead of programming them and
  - understanding how humans think
- each such formalisms allows to formalise certain **abstractions** from the "real world"
- different KR formalisms vary w.r.t.
  - how they represent knowledge (in graphs, formulae, frames, pictures, etc)
  - their basic assumptions/their atomic constructs (propositional variables, time points, objects and classes, etc.)
  - their reasoning services (what to infer from explicitly stated knowledge)
  - what they are good at/what they were designed for (e.g., for solving planning problems, reasoning about spatial knowledge, handling terminological knowledge, etc.)

Logic – the first KR formalism?

Logic is definitely a formalism that

- can be used to represent knowledge and provides reasoning services
- developed by ancient Greeks, e.g., Parmenides, Platon, and Aristoteles as a method for modelling "correct human reasoning", i.e., in rhetorics
- concerned with
  - statements, formulae,
  - their form (syntax),
  - their meaning (semantics), and
  - their formal relation (e.g., implication)
- Boole (1815-1864) unified
  - algebra (terms, functions, etc) with
  - truth functional logic
- Frege and Gödel: fathers of modern **first order predicate logic**
- later more

Logic – the first KR formalism?

Logic is difficult for human consumption

- e.g., how long does it take you to read

$$\forall x \exists y \forall z((r(x,y) \wedge s(y,z) \Rightarrow (\neg s(a,y) \vee r(x,z)))$$

- or to check that it is equivalent to

$$\forall x \exists y \forall z((r(x,z) \vee \neg r(x,y) \vee \neg s(y,z) \vee \neg s(a,y))$$

Early KR formalisms

Most of the early KR formalisms were **graphical** because graphics are

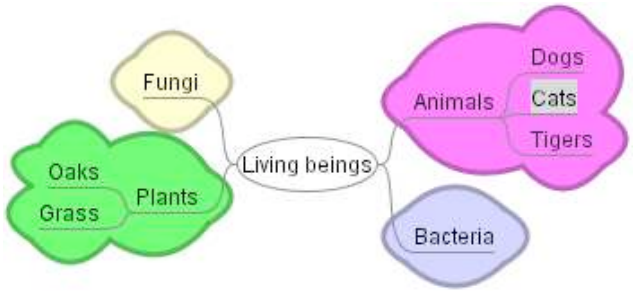
- easier to grasp: "a picture says more than thousand words"
- close to the way in which knowledge is represented in human beings (?)

Most graphical KR formalisms represent knowledge as

- **graphs** with
  - vertexes, possibly labelled, mostly representing concepts, classes, individuals, etc.
  - edges, possibly labelled, mostly representing properties, relationships, etc.

Next: many examples of such graphics!

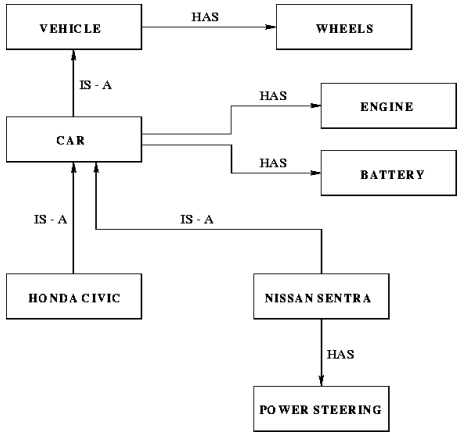
A MindMap



Why is this called a Mindmap?

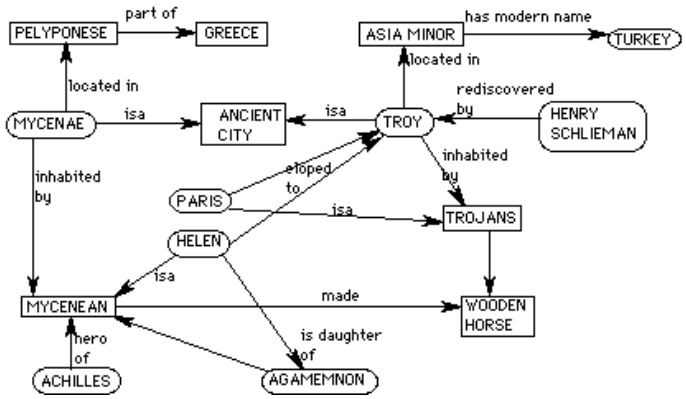
What does it represent? What does it say?

A Semantic Network



What does it represent? What does it say?

Another Semantic Network



What does it represent? What does it say?

Does Helen inhabit Mycenae?

What are differences between the previous SN and this one?

Semantic Networks: the IS-A relation

In semantic network and related formalism, the **is-a relationship** plays a central role, e.g., "an elephant is a mammal" or "a truck is a vehicle"

What does it mean if we find an is-a relationship between two nodes, from A to B?

- if A and B are concepts/classes, does is-a translate to
  - each instance of A is also an instance of B? E.g., each square is a rectangle
  - by default/normally As are Bs? E.g. normally birds are flying animals?
  - As inherit all properties of B—if not stated otherwise? E.g., white elephants are elephants, but they are not grey, but white
  - ...?

Semantic Networks: the IS-A relation (continued)

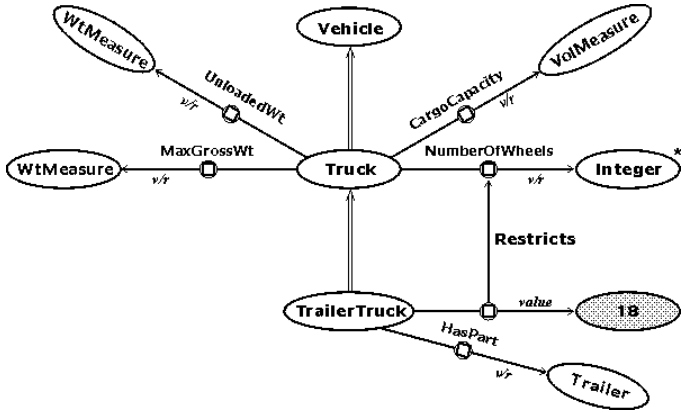
- if $A_1, \ldots, A_k$ are all sub-classes of a class B, does this imply that
  - A_i and A_j are **disjoint** for each $1 \leq i < j \leq n$, i.e., cannot have a common instance?
  - that each instance of B is an instance of some A_i, i.e., the A_j **cover** B?
  - none or both of the above?
- if A is an individual and B is a class: A is an instance of B?

Many SN formalisms are/were not very clear about that—KL-ONE was developed to make such questions concerning the **meaning** of representations more precise

⇝ KL-ONE came with a **well-defined semantics**

Syntax: what are well-formed structures/terms/formulae/graphs?
Semantics: what is **the meaning** of such a well-formed structure?
E.g., what are its **instances**? What are its **models**?
When does a statement **hold** in an interpretation?
When is a statement **true**? **satisfiable**? **valid**?

A KL-ONE knowledge base



What does it represent? What does it say?

What are differences between the previous SN and this one?

KL-ONE

For KL-ONE, the developers Brachmann and Schmolze

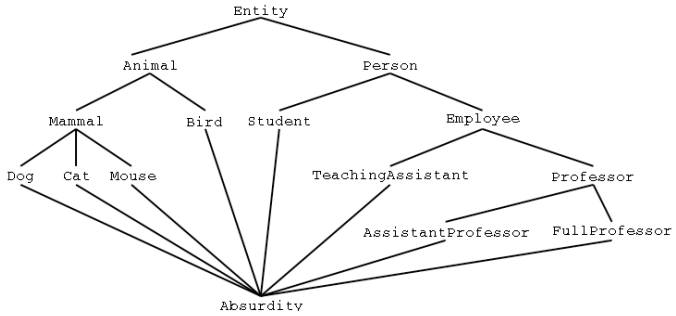
- explained exactly what it means to have a certain link between to nodes
- e.g., an is-a link (double-arrow) between A and B is read as "each instance of A is also an instance of B"
- decided to distinguish knowledge about concept, terms, etc. from knowledge about individuals, objects, etc.
- decided that finding implicit **is-a** relationships would be a useful reasoning service

Other formalisms were targeted towards other readings of is-a links: **non-monotonic inheritance networks**

A KR formalism is **monotonic** if "the more knowledge is given explicitly, the more can be derived"

In a **non-monotonic** one, from the facts that Tweety is a bird and birds can fly, we can deduce that Tweety can fly. However, if one learns later the Tweety is a Penguin,...

Non-monotonic inheritance versus monotonic inheritance



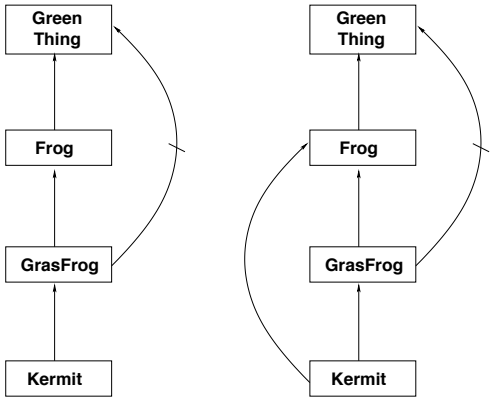
What is the relation between Mammals and Birds?

What is the relation between Students and Employees?

How does the "children" of Mammal differ from those of Professor?

Is this a monotonic network?

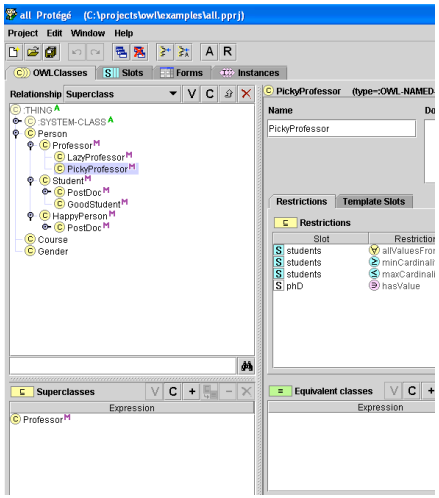
Non-monotonic inheritance versus monotonic inheritance



In non-monotonic network, is-a links can be **cancelled**, e.g., Grasfrogs are not green.

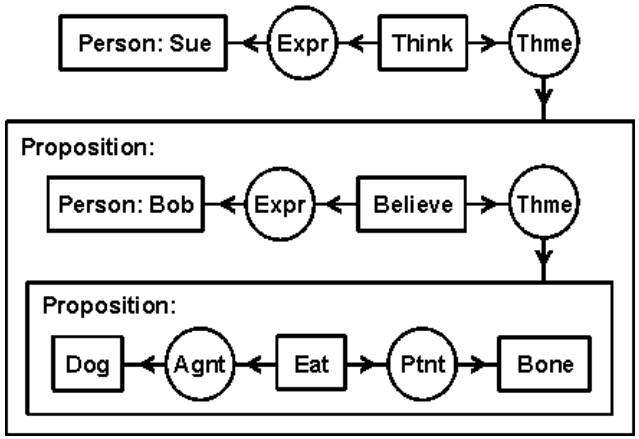
For each of the networks: what is Kermit?
Many variants of these cancellations were developed.

Other KR formalisms: Frame-based systems



Frame-based systems such as Protege: class-based, properties, classes are defined by stating super-classes and restricting the "fillers" of properties

Other KR formalisms: Conceptual Graphs



Conceptual Graphs, a "graphical syntax for first order logic"
widely used for natural language processing

The choice of a good KR formalisms

- there are many different KR formalisms around that
  - provide different expressive power
  - come with different basic, atomic entities
  - provide different reasoning services
  - were designed for different applications
  - have different interfaces pleasing different tastes
  - were made with different "religious" beliefs
  - ...
- some KR formalisms claim that they are "good for all applications"...
- we claim that different applications might use different formalisms, and that certain applications should even use several formalisms!

First Order Logic [2, supervised Lab]

Uli Sattler

General Remarks

For this session on First Order Logic,

- we assume that you have
  - some prior knowledge in logic (see course pre-requisites) and
  - completed the pre-course work.
- we will be rather fast, merely repeating the important notions of FOL

FOL plays an important role in KR because many KR formalisms

- can be understood by translating them into FOL
- are notational variants of (fragments/extensions) of FOL
- are closely related to (fragments/extensions) of FOL

Thus FOL can be viewed as a "common language" to talk about KR formalisms

FOL: Syntax — Terms

Signature: let $\mathcal{P}$, $\mathcal{F}$, $\mathcal{X}$ be pairwise disjoint sets of **predicate**, **function**, and **variable symbols**.
Each $P \in \mathcal{P}$ and each $f \in \mathcal{F}$ comes with an **arity** $n \in \mathbb{N}$.
Then $\Sigma = (\mathcal{P}, \mathcal{X}, \mathcal{F})$ is called a **signature**.

Examples: for predicates: Dog[1], likes[2], parents-of[3]

Remarks: we call functions of arity 0 **constants**
sometimes, $\mathcal{P}$ contains a binary equality symbol, sometimes not (ask!)

Terms: Let $\Sigma = (\mathcal{P}, \mathcal{X}, \mathcal{F})$ be a signature. The **terms over** Σ are **inductively** defined as follows:

- each constant and each variable is a term,
- if $f \in \mathcal{F}$ is a function of arity n and $t_1, \ldots, t_n$ are terms, then $f(t_1, \ldots, t_n)$ is also a term.

FOL: Syntax — Formulae

Formulae: The **formulae over** $\Sigma = (\mathcal{P}, \mathcal{X}, \mathcal{F})$ are **inductively** defined as follows:

- if $P \in \mathcal{P}$ is n-ary and $t_1, \ldots, t_n$ are terms, then $P(t_1, \ldots, t_n)$ is a formula,
- if ϕ and ψ are formulae, then so are $\neg\phi$, $\phi \wedge \psi$, and $\phi \vee \psi$, and
- if ϕ is a formula and $x \in \mathcal{X}$ a variable, then $\forall x.\phi$ and $\exists x.\phi$ are formulae

Useful notions:

- **sub-formulae sub**(ϕ) of a formula ϕ: the set of all sub-strings of ϕ being formulae
- if $Qx.\psi \in$ **sub**(ϕ) for a quantifier $Q \in \{\forall, \exists\}$, then we call ψ the **scope** of Q
- x **occurs in the scope** of a quantifier $Q \in \{\forall, \exists\}$ in ϕ if $Qx.\psi \in$ **sub**(ϕ)
- x **occurs free** in ϕ: if x occurs out-side the scope of a quantifier in ϕ
- a formula is **closed** if it contains no free variables

Let's see some examples of terms, formulae, free and bound variables!

FOL: Semantics — Interpretations

An **interpretation** $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a

- non-empty set $\Delta^{\mathcal{I}}$, the **interpretation domain**, and
- a function $\cdot^{\mathcal{I}}$ that maps
  - each $P \in \mathcal{P}$ of arity n to an n-ary relation $P^{\mathcal{I}} \subseteq (\Delta^{\mathcal{I}})^n$
  - each $f \in \mathcal{F}$ of arity n to an n-ary function $f^{\mathcal{I}} : (\Delta^{\mathcal{I}})^n \to \Delta^{\mathcal{I}}$

Remark: calling a 0-ary function f a "constant" is ok since $f^{\mathcal{I}} : \emptyset \to \Delta^{\mathcal{I}}$,
i.e., $f^{\mathcal{I}}$ stands for a **single element** in $\Delta^{\mathcal{I}}$

Let's see some example interpretations!

FOL: Semantics — How terms are interpreted

In the following, we use $\mathcal{I}$ for an interpretation, $\Sigma = (\mathcal{P}, \mathcal{X}, \mathcal{F})$ for a signature, x for variables, etc.

$\mathcal{I}$-assignment: A function $\alpha : \mathcal{X} \to \Delta^{\mathcal{I}}$ is called an **$\mathcal{I}$-assignment**.

Interpretation of terms: for α an $\mathcal{I}$-assignment, the interpretation $t^{\alpha,\mathcal{I}}$ of terms is inductively defined as follows:

$$\text{if } t = x \in \mathcal{X}, \text{ then } t^{\alpha,\mathcal{I}} = x^{\alpha,\mathcal{I}} = \alpha(x),$$
$$\text{if } t = f(t_1, \ldots, t_n), \text{ then } t^{\alpha,\mathcal{I}} = f^{\mathcal{I}}(t_1^{\alpha,\mathcal{I}}, \ldots, t_n^{\alpha,\mathcal{I}})$$

x-variants: two $\mathcal{I}$-assignments α, α' are **x-variants** of each other if $\alpha(y) = \alpha'(y)$ for all $y \in \mathcal{X}$ with $y \neq x$.

Make sure you understand what $t^{\alpha,\mathcal{I}}$ stands for!

FOL: Semantics — How formulae are interpreted

$\models$ For $\mathcal{I}$ an interpretation, α an $\mathcal{I}$-assignment,

we define the relation $\models$ by induction on the structure of formulae as follows

(where $\mathcal{I}, \alpha \models \phi$ is read as "ϕ **holds in** $\mathcal{I}$ **under** α"):

$\mathcal{I}, \alpha \models P(t_1, \ldots, t_n)$ iff $(t_1^{\alpha,\mathcal{I}}, \ldots, t_n^{\alpha,I}) \in P^{\mathcal{I}}$

$\mathcal{I}, \alpha \models t_1 = t_2$ iff $t_1^{\alpha,\mathcal{I}} = t_2^{\alpha,\mathcal{I}}$

$\mathcal{I}, \alpha \models \neg\phi$ iff it is **not** the case that $\mathcal{I}, \alpha \models \phi$

$\mathcal{I}, \alpha \models \phi \wedge \psi$ iff $\mathcal{I}, \alpha \models \phi$ **and** $\mathcal{I}, \alpha \models \psi$

$\mathcal{I}, \alpha \models \phi \vee \psi$ iff $\mathcal{I}, \alpha \models \phi$ **or** $\mathcal{I}, \alpha \models \psi$

$\mathcal{I}, \alpha \models \exists x.\phi$ iff there is an x-variant α' of α with $\mathcal{I}, \alpha' \models \phi$

$\mathcal{I}, \alpha \models \forall x.\phi$ iff for all x-variants α' of α, we have $\mathcal{I}, \alpha' \models \phi$

Consider the previous example interpretation $\mathcal{I}$ and some $\mathcal{I}$-assignment and check whether $\mathcal{I}, \alpha \models \phi$ for each ϕ in

$$P(a), P(x), \exists x.P(x), \forall x.P(x), \forall x.\exists y.Q(x,y), \exists y.\forall x.Q(x,y)$$

FOL: Semantics and Inference Problems

Remarks: (a) we use the following abbreviations

$\top$ for $P(a) \vee \neg P(a)$ (or any other tautology)

$\bot$ for $\neg\top$

$\phi \Rightarrow \psi$ for $\neg\phi \vee \psi$

$\phi \Leftrightarrow \psi$ for $(\phi \Rightarrow \psi) \wedge (\psi \Rightarrow \phi)$

(b) for closed formulae, the relation $\models$ does not depend on assignments:
for ϕ closed, we have $\mathcal{I}, \alpha \models \phi$ iff $\mathcal{I}, \alpha' \models \phi$.
Hence we often do not mention assignments α.

Finally: Let Γ be a set of closed formulae. We say that Γ

- is **true in** $\mathcal{I}$ (written $\mathcal{I} \models \Gamma$) if $\mathcal{I} \models \phi$ for each $\phi \in \Gamma$, in this case, $\mathcal{I}$ is called a **model of** Γ
- is **satisfiable** if there exists some $\mathcal{I}$ such that $\mathcal{I} \models \Gamma$
- is **valid** (or a tautology) if, for all $\mathcal{I}$, we have $\mathcal{I} \models \Gamma$
- has ψ as **a consequence** (written $\Gamma \models \psi$) if $\mathcal{I} \models \Gamma$ implies $\mathcal{I} \models \psi$, for each $\mathcal{I}$

FOL as a KR formalism: Examples

- Every natural number is the sum of two natural numbers:

$$\forall x.NN(x) \Rightarrow \exists y, z.(NN(y) \wedge NN(z) \wedge ADD(x,y,z))$$

- everybody loves somebody

$$\forall x.\exists y.loves(x,y)$$

- there is somebody who loves everybody

$$\exists x.\forall y.loves(x,y)$$

- nobody loves anybody who only loves him- or herself (intended reading):

$$\forall x.\Big((\forall y.loves(x,y) \Rightarrow x = y) \Rightarrow \neg\exists z.(loves(z,x) \wedge x \neq z)\Big)$$

- P is a linear order (an anti-symmetric, total, and transitive relation):

$$\begin{aligned}\phi_{tot}^{P} := \forall x.(\ & \neg P(x,x) \wedge \\ & [\forall y.P(x,y) \vee P(y,x) \vee x = y] \wedge \\ & [\forall y,z.(P(x,y) \wedge P(y,z)) \Rightarrow P(x,z)])\end{aligned}$$

FOL as a KR formalism

Clearly, we can represent some knowledge about an application domain in FOL,
i.e., we construct a set of formulae Γ to capture our knowledge about this domain

- which formulae to put into Γ?
  - each model $\mathcal{I}$ of Γ should conform to our intuition and
  - if an $\mathcal{I}$ is not a model of Γ, then $\mathcal{I}$ should not conform to our intuition
  - adding a tautology to Γ is useless/redundant: it does not change Γ's models

  How difficult is it to verify these points? How many interpretations are there?
- what does it mean to "conform to our intuition"? E.g., if $a \in P^{\mathcal{I}}$ and $a \in Q^{\mathcal{I}}$, then we should not think that P and Q are disjoint
- if $\Gamma \models \psi$, then ψ should "conform to our intuition"
- if Γ is unsatisfiable, then it does not have any model, and thus it should be useless
- we can deduce new, **implicit** knowledge from the one **explicitly** stated in Γ: test for $\Gamma \models \phi$, i.e., $\Gamma \models$ **Happy**(**John**) or $\Gamma \models \forall x.$**Child**$(x) \Rightarrow$ **Human**(x)

FOL as a KR formalism

Clearly, checking the satisfiability/validity/consequence status of a "knowledge base" Γ can help the designer of a knowledge base to build one that indeed "conforms to his/her intuition" (later more about that).

In the remainder of this session, we will

- see an algorithm that can be used for such checks. Before this, we will
- see how these **reasoning problems** are related to each other, and
- discuss a normal form of formulae that will make our life easier

Theorem 1: Let Γ be a **finite** set of closed formulae and ψ a closed formula.

1. Γ is valid (satisfiable) iff $\bigwedge_{\phi\in\Gamma}\phi$ is valid (satisfiable)
2. ϕ is valid iff $\neg\phi$ is not satisfiable
3. ψ is a consequence of ϕ iff $\phi \wedge \{\neg\psi\}$ is not satisfiable
4. ψ is satisfiable iff $\bot$ is not a consequence of ψ

FOL — Normal Form I

As a consequence of Theorem 1, we can reduce all reasoning problems from satisfiability, validity, and consequence to (the negation of) each other — which is nice, because we can concentrate on one we/I like most: satisfiability!

Equivalence: ψ is equivalent to ϕ (written $\psi \equiv \phi$) iff $\{\psi \Leftrightarrow \phi\}$ is valid.

Negation Normal Form: a formula is in **NNF** if negation occurs only in front of atomic formulae (predicates over terms)

Lemma: Each formula can be re-written into an equivalent one in NNF using a combination of deMorgan's laws and the duality between existential and universal quantification.

Let's see some examples!

FOL — Tableaux

So, without loss of generality, we restrict our attention to **closed** formulae in **NNF**.

Substitution: A **substitution** σ is a mapping from **variables to terms** with a finite domain.
$\sigma(\psi)$ is the formula obtained by replacing each **free** occurrence of a variable x in ψ that is in the domain of σ with $\sigma(x)$.

Example: for $\sigma = \{x \mapsto f(a),\ y \mapsto g(z,w)\}$,
$\sigma(\,P(x) \wedge \forall x.R(x,y)) =$
$P(f(a)) \wedge \forall x.R(x, g(z,w))$

Tableau for ψ: A **tableau for** ψ is a binary tree whose nodes are labelled with sub-formulae of ψ, the root being labelled with ψ.

For a leaf node v in a tableau, we define **its branch** $B(v)$ as follows:

$B(v) = \{\phi \mid \phi$ is the label of a node in the path from the root node to $v\}$

FOL — Tableau algorithm: the rules

In the following, $B(v)$ denotes a branch for a tableau with v a leaf node.

Closed: a **branch** $B(v)$ is **closed** if it contains a formula and its negation.
No rules are applied to closed branches.

A **tableau** is **closed** if all its branches are closed.

Tableau Rules work on tableaux and append new nodes (1-2) to leaf nodes. They come in two forms, and are read as follows:

$$\frac{\psi}{\begin{array}{l}\phi_1\\ \vdots\\ \phi_n\end{array}}$$ if $\psi \in B(v)$, you can add a successor node to v labelled with one of the ϕ_i

$$\frac{\psi}{\phi_1 \mid \phi_2}$$ if $\psi \in B(v)$, you can add 2 succ. nodes to v labelled with ϕ_1 and ϕ_2, resp.

FOL — Tableau algorithm: the rules

These are the four tableau rules for FOL—
without equality, in NNF, applicable to non-closed branches $B(v)$:

$$\frac{\psi_1 \wedge \psi_2}{\begin{array}{c}\psi_1\\ \psi_2\end{array}} \qquad \frac{\exists x.\psi}{\{x \mapsto t\}\psi} \quad \text{for some term } t \text{ not occurring in } B(v)$$

$$\frac{\psi_1 \vee \psi_2}{\psi_1 \mid \psi_2} \qquad \frac{\forall x.\psi}{\{x \mapsto t\}\psi} \quad \text{for some term } t$$

FOL — Tableau algorithm: Examples

Example: apply the tableau algorithm to the following formulae:

$$A(a) \wedge \forall x.\Big(B(x) \wedge \big(C(x) \vee \exists y.S(x,y)\big)\Big)$$

Important: if you were asked to implement the tableau algorithm, you had to decide on

- a nice **data-structure** for branches and formulae
- a **strategy** for the **order** in which the rules are applied
  here, it is **"don't-care" non-deterministic**
- a **strategy** for the choice of terms in the $\forall$- and the $\exists$-rule
- a **strategy** for the choice of ϕ_i in the $\wedge$-rule
- a **strategy** for how to build/search the tableau
  it is a tree, hence e.g., we can use depth-first or breadth-first, or something else
- etc.

FOL — Tableau algorithm: Properties and Examples

Answer behaviour: the tableau algorithm answers "ψ **is not satisfiable**" if,
when started with ψ,
it **can** generate a closed tableau

Examples: we apply the TA to these formulae and describe its answer:

1. $\phi_1 = \exists x.R(a,x) \vee (A(a) \vee \forall y.\neg R(a,y))$
   yields a tableau with a "non-closable" branch
   (from which we can build a model for ϕ_1)
2. $\phi_2 = \exists x.R(a,x) \wedge (A(a) \vee \forall y.\neg R(a,y))$
   yields a tableau with a "non-closable" branch
   (from which we can build a model for ϕ_2)
3. $\phi_3 = \exists x.R(a,x) \wedge (A(a) \wedge \forall y.\neg R(a,y))$
   can lead to a closed tableau $\rightsquigarrow$ ϕ_3 is unsatisfiable, i.e., $\neg\phi_3$ is valid

FOL — Tableau algorithm: Properties and Examples

4. $A(a) \wedge \exists x.\neg A(x)$ yields a tableau with a single non-closed branch
   $$\{A(a) \wedge \exists x.\neg A(x),\ A(a),\ \exists x.\neg A(x), \neg A(b)\},$$
   we can stop since replacing x with other, **new** terms wouldn't close this branch
   $\rightsquigarrow$ satisfiable
5. $A(a) \wedge \forall x.\neg A(x)$ **can** yield a tableau with a single **closed** branch
   $$\{A(a) \wedge \exists x.\neg A(x),\ A(a),\ \exists x.\neg A(x), \neg A(a)\},$$
   $\rightsquigarrow$ **unsatisfiable**
6. $A(a) \wedge \forall x.\exists y.S(x,y)$ yields a tableau with a single non-closed branch
   $$\{A(a) \wedge \forall x.\exists y.S(x,y),\ A(a),\ \forall x.\exists y.S(x,y),\ \exists y.S(a,y),\ S(a,b),\ \exists y.S(b,y),\ S(b,b_1),\ \exists y.S(b_1,y),\ S(b_1,b_2),\ \exists y.S(b_2,y),\ S(b_2,b_3),\ldots\}$$
   that will never close (how do we know?), no matter how we substitute/which rules we apply $\rightsquigarrow$ we can answer **satisfiable**...but how can the algorithm tell?

FOL — Tableau algorithm: properties

What does the tableau algorithm do when started with ψ?

- does it terminate? **No:** e.g., consider $A(a) \wedge \forall x.\exists y.S(x,y)$
- does it lie? **No:** we can show that, if the tableau yields a closed tableau for ψ, then ψ is unsatisfiable $\rightsquigarrow$ **sound** for unsatisfiability/validity
- what happens if ψ is unsatisfiable? With a **fair strategy** for rule applications, it will eventually return "ψ is not satisfiable" $\rightsquigarrow$ **complete** for unsatisfiability/validity
- how is it used to test **validity** of ψ or whether ψ is a **consequence of** a (finite) Γ?
  Easy: use TA as "sub-program":
  - for validity of ψ, start TA with NNF of $\neg\psi$ and return "ψ is valid" if TA can generate closed tableau for $\neg\psi$, **returns "ψ is valid" iff ψ is indeed valid**
  - for $\Gamma \models \psi$, see coursework!

FOL — could we do better than the tableau algorithm?

As we have seen, the tableau algorithm does not terminate.

Obvious Question: can't we design a **terminating** algorithm for validity of FOL that is sound **and** complete?
I.e., one that returns "ψ is valid" iff ψ is indeed valid?

I.e., can't we design a **decision procedure** for validity of FOL formulae?

Sad Answer: No, we can not—validity of FOL formulae is **undecidable**

However, our tableau algorithm is a **semi-decision-procedure**:
if ψ is valid, then TA terminates on (NNF of) $\neg\psi$ (with closed tableau)
thus the validity problem is **semi-decidable**,

and the set of valid formulae is **recursively enumerable**

FOL — could we do better than the tableau algorithm?

Reactions:

- we stop bothering with logics: they are all undecidable anyway ;)
- we implement TAs and similar algorithms and optimise them $\rightsquigarrow$ theorem prover/automated reasoning
- we investigate **fragments** of FOL: try to find **expressive** fragments that are decidable, e.g., modal and description logics

before we do this, we will see **why** FOL is undecidable

FOL — why is validity/satisfiability of FOL so complex/undecidable?

Naive Answer: because there are FOL formulae that only have infinite domains, e.g.,

$$\phi^{P}_{tot} \wedge \forall x.\exists y.(P(x,y))$$

Hence a satisfiability decision procedure would need to check for existence of an **infinite model**

However, there are logics with similar expressive power that **are** decidable...

Precise Answer: take an **undecidable problem** P and reduce P to satisfiability of FOL formula, i.e.,

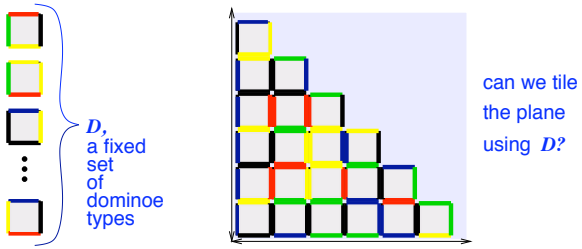
choose undecidable problem $P \subseteq M$ and define computable function $\pi : M \longrightarrow \Sigma^*$ such that

$$p \in P \text{ iff } \pi(p) \text{ is satisfiable}$$

FOL — why is validity/satisfiability of FOL undecidable? (ctd)

Consequence of undecidable problem $P \subseteq M$ and computable function $\pi : M \to \Sigma^*$ with $p \in P$ iff $\pi(p)$ is satisfiable:
if we had a decision procedure for FOL satisfiability, we could use it to decide P via π—which we know we cannot,
hence FOL satisfiability is undecidable

Domino problem: we will use the undecidable domino problem as P:



FOL — why is validity/satisfiability of FOL undecidable? (ctd)

Definition: A **domino system** $\mathcal{D} = (D, H, V)$

- set of domino **types** $D = \{D_1, \ldots, D_d\}$, and
- horizontal and vertical **matching conditions** $H \subseteq D \times D$ and $V \subseteq D \times D$

A **tiling** for $\mathcal{D}$ is a (total) function:

$$t : \mathbb{N} \times \mathbb{N} \to D \text{ such that}$$
$$\langle t(m,n), t(m+1,n)\rangle \in H \text{ and}$$
$$\langle t(m,n), t(m,n+1)\rangle \in V$$

Domino problem: given $\mathcal{D}$, has $\mathcal{D}$ a tiling?

It is well-known that this problem is undecidable [Berger66]

FOL — why is validity/satisfiability of FOL undecidable? (ctd)

Reducing the domino problem to satisfiability of FOL formulae $\rightsquigarrow$ **4 tasks:**

so we translate $\mathcal{D} = (\{D_1, \ldots, D_d\}, H, V)$ into

$\pi(\mathcal{D}) = \delta_1 \wedge \delta_2 \wedge \delta_3 \wedge \delta_4$

① each object carries **exactly one domino type** D_i
$\rightsquigarrow$ use unary predicate symbol D_i for each domino type and

$$\delta_1 = \forall x. \bigvee_{1 \le i \le d} (D_i(x) \wedge \bigwedge_{j \neq i} \neg D_j(x))$$

② we have relations X, Y for the horizontal and vertical axis, and
$N_X(\cdot,\cdot)$, $N_Y(\cdot,\cdot)$ for the direct horizontal and vertical successors:

$$\begin{aligned}\delta_2 = \; & \phi^X_{tot} \wedge \forall x.\exists y.N_X(x,y) \wedge \\ & [\forall x,y.N_X(x,y) \Leftrightarrow (X(x,y) \wedge \neg\exists z.(X(x,z) \wedge X(z,y)))] \wedge \\ & \phi^Y_{tot} \wedge \forall x.\exists y.N_Y(x,y) \wedge \\ & [\forall x,y.N_Y(x,y) \Leftrightarrow (Y(x,y) \wedge \neg\exists z.(Y(x,z) \wedge Y(z,y)))]\end{aligned}$$

FOL — why is validity/satisfiability of FOL undecidable? (ctd)

Please note that N_X and N_Y behave like total functions,
i.e., $N_X(a,b)$ and $N_X(a,b')$ implies $b = b'$.

③ for each node, ensure that its horizontal-vertical-successor **coincides** with its vertical-horizontal-successor

$$\delta_3 = \forall x,y,z.(N_X(x,y) \wedge N_Y(y,z)) \Rightarrow (\exists y'.N_Y(x,y') \wedge N_X(y',z))$$

④ each element satisfies the **horizontal/vertical matching conditions**:

$$\delta_4 = \forall x. \bigwedge_{1 \le i \le d} \Big(D_i(x) \Rightarrow \big((\forall y.N_X(x,y) \Rightarrow \textstyle\bigvee_{(D_i,D_j) \in H} D_j(y)) \wedge (\forall y.N_Y(x,y) \Rightarrow \bigvee_{(D_i,D_j) \in H} D_j(y))\big)\Big)$$

FOL — why is validity/satisfiability of FOL undecidable? (ctd)

Harvest: $\underbrace{\pi(\mathcal{D})}_{\delta_1 \wedge \delta_2 \wedge \delta_3 \wedge \delta_4}$ is satisfiable iff $\mathcal{D}$ has a tiling

since the domino problem is undecidable, this implies undecidability of the satisfiability of FOL formulae

Summary

In this section, we have

- repeated important concepts of **first order logic**,
- discussed the **inference problems** satisfiability, validity, and consequence, and their inter-relationship,
- seen a sound and complete **algorithm** for the satisfiability of FOL formulae, and
- seen why satisfiability of FOL formulae is undecidable.