Artificial Intelligence

An introduction

Maik Kschischo

Institute for Computer Science
University of Koblenz

Table of Contents

Chap. 1 Overview and history of AI

Chap. 2 Propositional logic

Chap. 3 First order logic

Chap. 4 Applications and limitations of logic in AI

Applications

Selected examples

- Planning: Plan the actions of a robotic
- Knowledge representation and reasoning
- Automatic theorem proofing in Mathematics

Planning

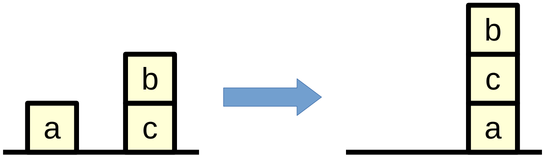


Figure: A planning task for a robot.

Planning tasks are characterized by

- an initial situation,
- actions, which can change the current situation
- a target situation.

STRIPS I

Stanford Research Institute Problem Solver

- describes a situation $\mathcal{S}$ as a PL1 expression
- actions change a situation $\mathcal{S}$ by deleting and adding clauses in $\mathcal{S}$
- actions can only be performed, if the situation $\mathcal{S}$ fulfills certain constraints $\mathcal{C}$

Actions:

Name of the action:(parameters of the action)

Constraint:	PL1 expression $\mathcal{C}$	to be fulfilled by current$\mathcal{S}$
Action:		
	add:	Clauses to be added to $\mathcal{S}$
	delete:	Clauses to be deleted from $\mathcal{S}$

STRIPS II

Stanford Research Institute Problem Solver

The constraint of the action is fulfilled if $\mathcal{S} \vDash \mathcal{C}$.

- The constraints $\mathcal{C}$ are described by a set of clauses to be fulfilled in the situation $\mathcal{S}$.
- To proof $\mathcal{S} \vDash \mathcal{C}$ one adds $\neg\mathcal{C}$ to the set of clauses $\mathcal{S}$ and generates an empty clause (proof by contradiction).

Planning the actions of a robot I

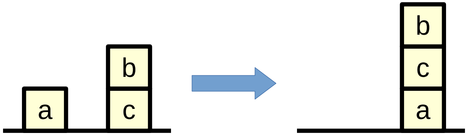


Figure: A planning task for a robot.

The planning task can be described in PL1 by initial and the desired final state

$$\begin{array}{c} Bottom(a) \\ Bottom(c) \\ Ontop(b,c) \\ Free(a) \\ Free(b) \end{array} \longrightarrow \begin{array}{c} Bottom(a) \\ Ontop(c,a) \\ Ontop(b,c) \\ Free(b) \end{array}$$

Planning the actions of a robot II

The robot can perform two different actions

1. "stack" block 1 on top of block 2
2. "lay down" of block 1, which is currently on top of block 2

Stacking$(Block_1, Block_2)$

Constraint:	$Free(Block1)$	
	$Free(Block2)$	
Action:		
	delete:	$Free(Block2)$
		$Bottom(Block1)$
	add:	$Ontop(Block1, Block2)$

Planning the actions of a robot III

$Laydown(Block_1, Block_2)$

Constraint:	$Free(Block1)$	
	$Ontop(Block1, Block2)$	
Action:		
	delete:	$Ontop(Block1, Block2)$
	add:	$Free(Block2)$
		$Bottom(Block1)$

Compare the start and target situation in our example

Planning the actions of a robot IV

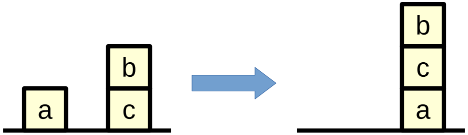


Figure: A planning task for a robot.

$$\begin{array}{c} Bottom(a) \\ Bottom(c) \\ Ontop(b,c) \\ Free(a) \\ Free(b) \end{array} \quad \longrightarrow \quad \begin{array}{c} Bottom(a) \\ Ontop(c,a) \\ Ontop(b,c) \\ Free(b) \\ \end{array}$$

Planning the actions of a robot V

1. We choose as the first action

$Laydown(b, c)$		
Constraint:	$Free(b)$	
	$Ontop(b, c)$	
Action:		
	delete:	$Ontop(b, c)$
	add:	$Free(c)$
		$Bottom(b)$

Planning the actions of a robot VI

2. To proof $\mathcal{S} \models \mathcal{C}$ we add the negation of $\mathcal{C} \equiv \mathit{Free}(b) \land \mathit{Ontop}(b, c)$ to the clauses describing the initiaial state

$$\begin{aligned}
&\neg \mathit{Free}(b) \lor \neg \mathit{Ontop}(b, c)\\
&\mathit{Bottom}(a)\\
&\mathit{Bottom}(c)\\
&\mathit{Ontop}(b, c)\\
&\mathit{Free}(a)\\
&\mathit{Free}(b)
\end{aligned}$$

Planning the actions of a robot VII

Here, we can easily derive the empty clause. After performing the action, the new situation is given by

$$\begin{gathered}
Bottom(a)\\
Bottom(b)\\
Bottom(c)\\
Free(a)\\
Free(b)\\
Free(c)
\end{gathered}$$

Planning the actions of a robot VIII

3. Next, we put block c on top of block a, i.e. we perform the action

$$
\begin{array}{lll}
\multicolumn{3}{l}{\mathit{Stacking}(c, a)} \\
\text{Constraint:} & \mathit{Free}(c) & \\
 & \mathit{Free}(a) & \\
\text{Action:} & & \\
 & \text{delete:} & \mathit{Free}(a) \\
 & & \mathit{Bottom}(c) \\
 & \text{add:} & \mathit{Ontop}(c, a)
\end{array}
$$

Planning the actions of a robot IX

After checking the constraints (easy) and performing the action we obtain the situation

$$\begin{aligned}&\mathit{Bottom}(a)\\&\mathit{Bottom}(b)\\&\mathit{Ontop}(c, a)\\&\mathit{Free}(b)\\&\mathit{Free}(c)\end{aligned}$$

Planning the actions of a robot X

4. Now we put block b on top of block c to obtain the target situation

$$\begin{array}{llll}
\multicolumn{4}{l}{\mathit{Stacking}(b, c)} \\
& \text{Constraint:} & \mathit{Free}(b) & \\
& & \mathit{Free}(c) & \\
& \text{Action:} & & \\
& & \text{delete:} & \mathit{Free}(c) \\
& & & \mathit{Bottom}(b) \\
& & \text{add:} & \mathit{Ontop}(b, c)
\end{array}$$

Search

To solve a planning task, one needs to find a series of actions leading from the initial state to the desired final state.
Search graphs:

1. Nodes: represent the states
2. Edges: Are present if there exists an action leading from one state to the other.

See e.g. *Russel, P. and Norvig, S, chapter 3* for search algorithms.

Knowledge representation

Aims

Humans understand, reason and interprete knowledge. Based on their knowledge, they perform various actions in the real world.

Knowledge representation

- Knowledge representation and reasoning (KR, KRR) is the part of AI concerned with approaches to representing information about the real world so that a computer can understand and can utilize this knowledge to solve the complex real world problems.
  - diagnosis of a medical condition
  - finding a new engineering design
  - find interactions between genes and proteins
  - ...
- KR is not just storing data into some database, but it also enables an intelligent machine to reason on the basis of that knowledge.

Kinds of knowledge

to be represented in AI systems

Object:	All the facts about objects in our world domain (e.g. cars have wheels, electrical cars have batteries, combustion cars have fuel tanks,...).
Events:	Events are the actions which occur in our world (e.g. car drives, car breaks down, an accident,...).
Meta-knowledge:	It is knowledge about what we know.
Facts:	Facts are the truths about the real world and what we represent.
Knowledge base:	Collects the known facts regarding objects and events in the form of sentences (as a technical term).

Types of knowledge I

1. Declarative knowledge (descriptive knowledge):
   - is to know about something,
   - includes concepts, facts, and objects.
2. Procedural knowledge (imperative knowledge):
   - is about knowing how to do something,
   - includes rules, strategies, procedures, agendas, etc.,
   - is application dependent.
3. Heuristic knowledge:
   - represents knowledge of some experts in a field or subject,
   - rules of thumb based on previous experiences, awareness of approaches, and which are good to work with,
   - not guaranteed.
4. Structural knowledge:
   - describes relationships between various concepts such as kind of, part of, and grouping of something,

Types of knowledge II

- describes the relationship that exists between concepts or objects.

5. Meta-knowledge:
   - is knowledge about the other types of knowledge.

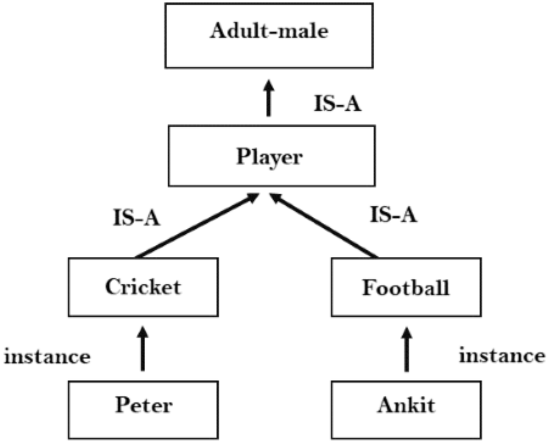
Approaches to Knowledge Representation I

- Relational data base:

Name	Age	Goals	Assists
Heinze	22	22	45
Palo	23	46	12
Müller	29	128	221

  - little opportunity for deductive inference
- Inheritable knowledge representation:
  - all data are stored into a hierarchy of classes
  - inheritable knowledge shows the relation between instance and class, and it is called instance relation
  - Class hierarchies can be represented graphically with arrows pointing from objects to values

Approaches to Knowledge Representation II



Approaches to Knowledge Representation III

- Inferential knowledge representation:
  - Knowledge is represented in the form of formal logics.
  - One can derive more facts from the inferential knowledge base.
  - Correctness is guaranteed.
  - Example: Marcus is a man. All men are mortal.

$$Man(Marcus)$$
$$\forall X Max(X) \Rightarrow Mortal(X)$$

- Procedural knowledge representation:
  - uses small programs and codes which describes how to do specific things, and how to proceed.
  - If-Then rules are used.
  - heuristic or domain-specific knowledge can be represented.

Techniques for knowledge representation

There are four important techniques for KR:

1. Logical Representation
2. Semantic Network Representation
3. Frame Representation
4. Production Rules

Logical knowledge representaion

Is based on logic (propositional or PL1)

Advantage:

- Enables us to do logical reasoning.

Disadvantages:

- Inference is often not efficient.
- Building knowledge bases requires familiarity with logic.

Semantic network representation I

- Semantic networks represent our knowledge in the form of graphical networks.
- Nodes represent objects and arcs describe the relationship between those objects.

Two types of relations:

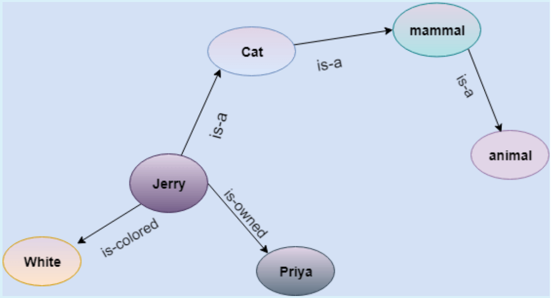
- IS-A relation (Inheritance)
- Kind-of-relation

Semantic network representation II

Example

Statements:

- Jerry is a cat.
- Jerry is a mammal.
- Jerry is owned by Priya.
- Jerry is brown colored.
- All Mammals are animals.



From https://www.javatpoint.com/ai-techniques-of-knowledge-representation

Semantic network representation III

Advantages:

- are a natural representation of knowledge,
- convey meaning in a transparent manner,
- simple and easily understandable.

Disadvantages:

- computational time might be prohibitive at runtime as we need to traverse the complete network tree to answer some questions,
- no quantifiers, e.g., $\forall, \exists$
- do not have any standard definition for the link names,
- logic reasoning and entailment are difficult to implement and the utility depends on the creator.

Frame Representations I

- A frame is a record like structure which consists of a collection of attributes and its values to describe an entity in the world.
- consists of a collection of slots and slot values.
- Slots have names and values which are called **facets**.

Facets:

- are features of frames which enable us to put constraints on the frames

Frame Representations II

Example

A robot should put screws into a chassis. There are two tasks

- "Put a 2×12 screw into drill hole 1"
- "Put a 4×30 screw into drill hole 2"

The positions and the types of screws are collected as frames

Object:	Screw 2×12
Diameter	2
Length	12
Position X	220 mm
Position Y	280 mm

Object:	Screw 4×30
Diameter	4
Length	30
Position X	100 mm
Position Y	150 mm

Frame Representations III

Example

In addition, the chassis is described as a frame

Object: Chassis		
	Drill hole: hole 1	
	Diameter:	2 mm
	X-Coordinate:	10 mm
	Y-Coordinate:	15 mm
	Drill holle: hole 2	
	Diameter:	4 mm
	X-Coordinate:	42 mm
	Y-Coordinate:	90 mm

Frame Representations IV

Advantages:

- Programming is easier by grouping the related data (object oriented programming).
- It is very easy to add slots for new attribute and relations.
- It is easy to include default data and to search for missing values.
- Frame representations are easy to understand and visualize.

Disadvantages:

- Inference mechanisms can not be easily processed.
- Frame representations can not easily proceed inference mechanisms.

Production Rules I

- Production rules consist of **(condition, action)** pairs which mean, "If condition then action".
- A KR by a production rule is defined by
  - the set of production rules,
  - working memory,
  - the recognize-act-cycle.
- Recognize-act-cycle
  - The agent checks for the condition and if the condition is met, the production rule fires and the corresponding action is carried out.
  - The condition part of the rule determines which rule may be applied to a problem.
- Working memory
  - contains the description of the current state
  - A rule can write knowledge to the working memory, which in turn may fire other rules.

Production Rules II

Example

- IF (at bus stop AND bus arrives) THEN action (get into the bus)
- IF (on the bus AND paid AND empty seat) THEN action (sit down).
- IF (on bus AND unpaid) THEN action (pay charges).
- IF (bus arrives at destination) THEN action (get down from the bus).

Production Rules III

Advantages:

- Production rules are expressed in natural language.
- Modularity, i.e it is easy to modify, remove or add a rule.

Disadvantages:

- No learning capabilities. The rules perform neither deductive nor inductive inference.
- Computational ineffciency, if many rules are active.

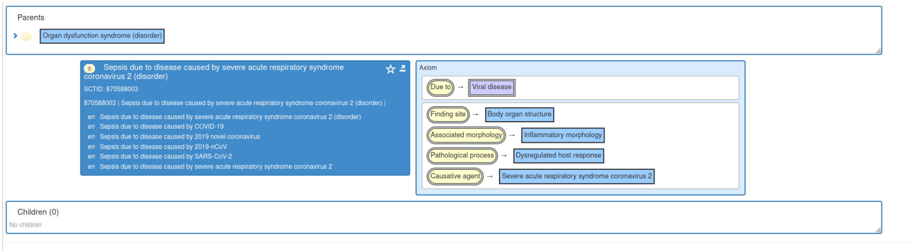
Ontologies I

What do we need to exchange knowledge?

- Syntax: Joint concepts, symbols and operators.
- Semantics: A joint understanding of the the meaning of a concept.
- Taxonomy: Classification of the concepts.
- Thesaurus: Relations between concepts
- Ontology: Rules and knowledge, about which rules are meaningful and allowed.

Example: I
SNOMED CT: a medical ontology

- Concept = medical teram
- Example for a particular concept: “Sepsis due to disease caused by respiratory syndrome coronavirus 2”


Parents
Organ dysfunction syndrome (disorder)

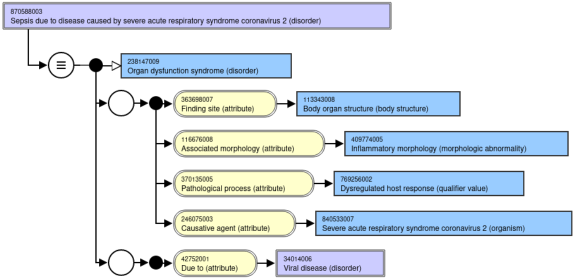
Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)
SCTID: 870588003
870588003 | Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder) |
en Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)
en Sepsis due to disease caused by COVID-19
en Sepsis due to disease caused by 2019 novel coronavirus
en Sepsis due to disease caused by 2019-nCoV
en Sepsis due to disease caused by SARS-CoV-2
en Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2

Axiom
Due to → Viral disease
Finding site → Body organ structure
Associated morphology → Inflammatory morphology
Pathological process → Dysregulated host response
Causative agent → Severe acute respiratory syndrome coronavirus 2

Children (0)
No children

Example: II

SNOMED CT: a medical ontology


870588003
Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)
≡
238147009
Organ dysfunction syndrome (disorder)
363698007
Finding site (attribute)
113343008
Body organ structure (body structure)
116676008
Associated morphology (attribute)
409774005
Inflammatory morphology (morphologic abnormality)
370135005
Pathological process (attribute)
769256002
Dysregulated host response (qualifier value)
246075003
Causative agent (attribute)
840533007
Severe acute respiratory syndrome coronavirus 2 (organism)
42752001
Due to (attribute)
34014006
Viral disease (disorder)

Example: III

SNOMED CT: a medical ontology

```
EquivalentClasses(
    :870588003 |Sepsis due to disease caused by severe acute respiratory syndrome coronavirus 2 (disorder)|
    ObjectIntersectionOf(
        :238147009 |Organ dysfunction syndrome (disorder)|
        ObjectSomeValuesFrom(
            :609096000 |Role group (attribute)|
            ObjectIntersectionOf(
                ObjectSomeValuesFrom(
                    :116676008 |Associated morphology (attribute)|
                    :409774005 |Inflammatory morphology (morphologic abnormality)|
                )
                ObjectSomeValuesFrom(
                    :246075003 |Causative agent (attribute)|
                    :840533007 |Severe acute respiratory syndrome coronavirus 2 (organism)|
                )
                ObjectSomeValuesFrom(
                    :363698007 |Finding site (attribute)|
                    :113343008 |Body organ structure (body structure)|
                )
                ObjectSomeValuesFrom(
                    :370135005 |Pathological process (attribute)|
                    :769256002 |Dysregulated host response (qualifier value)|
                )
            )
        )
        ObjectSomeValuesFrom(
            :609096000 |Role group (attribute)|
            ObjectSomeValuesFrom(
                :42752001 |Due to (attribute)|
                :34014006 |Viral disease (disorder)|
            )
        )
    )
)
```

Definition of an Ontology (Thomas R. Gruber, 1993)

Definition (Ontology)

An ontology is an explicit, formal specification of a shared conceptualization.

- Conceptualization: an abstract model.
- Domain: relevant terms (“concepts”) and relations between them (“relations”) in a certain fieldi of knowledge (”domain”).
- Explicit: all concepts must be defined.
- Formal: the machine must be able to process it.
- Shared: there is a consensus about the ontology.

Formal representation

Example

Informal representation

Lithium is a chemical element; it has symbol Li and atomic number 3. It is a soft, silvery-white alkali metal. Under standard conditions, it is the least dense metal and the least dense solid element. ...

(from Wikipedia)

Semi-Formal representatiom

Lithium

Name: $\langle String \rangle$

Symbol: $\langle String \rangle$

Atomic number $: Z = 3$

$\vdots$

More examples

- Disease Ontology
  - Represents diseases via common terms and via "is-a" relations
  - `https://www.disease-ontology.org/`
- Gene Ontology
  - Represents the knowledge about genes.
    - Molecular Function
    - Cellular Component
    - Biological Process
  - `khttps://geneontology.org/`

Representation of Ontologies I

Classes, relations und Instances

- Ontologies can be represented as **classes**, **relations** and **instances**.
- Classes are groups or sets of objects and represent the concepts of the ontology
- Classes are characterized by their **attributes**.
- Attributes are pairs of the form $(Name, Value)$

Representation of Ontologies II

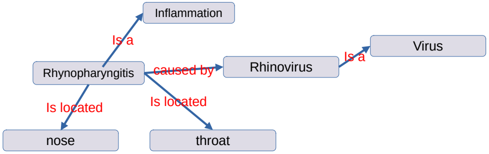
Classes, relations und Instances

Relations:

- relate different classes to each other
- are special attributes
- the values of a relation are objects of other classes

Informal representation

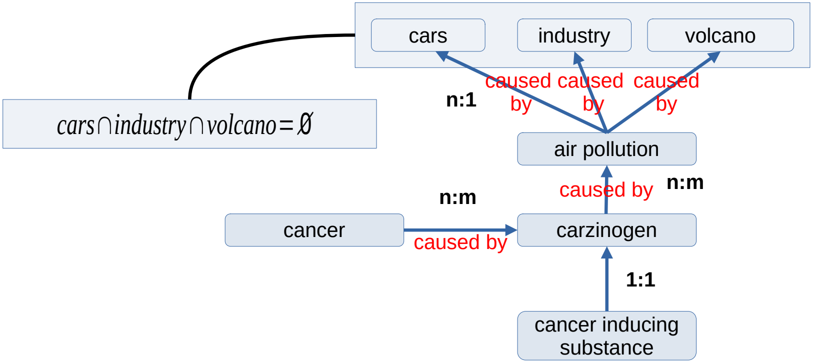
The common cold or the cold is a viral infectious disease of the upper respiratory tract that primarily affects the respiratory mucosa of the nose, throat, sinuses, and larynx.[6][8] Signs and symptoms may appear fewer than two days after exposure to the virus. These … (from Wikipedia)



Representation of Ontologies III

Classes, relations und Instances

Rules: Rules (constraints) define allowed values for relations and attributes.



Representation of Ontologies IV

Classes, relations und Instances

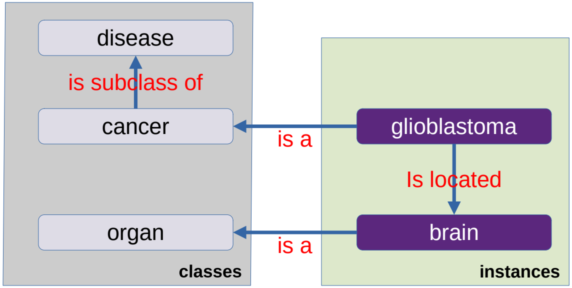
Propositions and Entailment

- Classes, relations and rules can be combined to form **propositions**.
- **Formal axioms:**
  - Formal axioms are special propositions, which are not expressed by the ontology itself.
  - Missing information is not declared as wrong, but unknown (**open world semantics**)
  - Example: "There is no clear evidence that COVID-19 is not a zoonosis."

Representation of Ontologies V

Classes, relations und Instances

Instances: describe an individuum in an ontology.



Ontologies and logic

Web Ontology Language (OWL) (`https://www.w3.org/OWL/`)

- is a family of knowledge representation languages for authoring ontologies.
- OWL implements **description logic**
- Description logic contains a well defined subset of PL1, which is decidable.
- There are inference engines for ontologies implemented in OWL.
- There is also an ontology editor called Protege to design ontologies.

Limitations of logic approaches to AI

- Search space problem.
- Decidability and incompleteness.
- Modeling uncertainty

The search space problem. I

- At every step of an deductive inference problem, there are many possibilities for the application of an inference rule.
- Search space problem: An explosive growth of the search space
- Worst case: All possible sequences of applying an inference rule must be tried

Figure: From Ertel, Introduction to AI, Springer, 2017, Fig. 4.1

The search space problem. II

Humans use heuristics to restrict the search space.

Heuristic proof controlling modules:

- Evaluate the various alternatives for the next step and then choose the alternative with the best rating.
- Resolution rule: the rating of the available clauses can be based on the number of positive literals, the complexity of the terms, etc., for every pair of resolvable clauses.
- Use a machine learning classifier:
  - Variables: attributes of all pairs of possible resolution steps
  - Classes: Positive resolutions, negative resolutions
  - Training data: Use an automatic theorem prover and save as training data for the positive class the variables for every proof found at every branch during the search. Negative examples are are given by branches not leading to a proof.

The search space problem. III

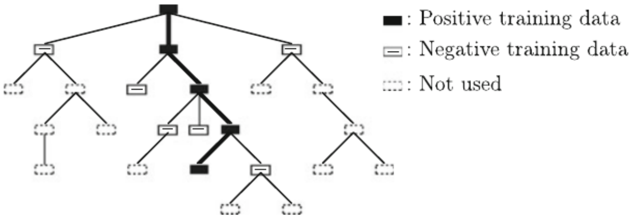


Figure: A search path of an automatic theorem proofer. Successful proofs follow paths labeled as positive examples and unsuccessful proofs follow paths labeled as negatives. From Ertel, Introduction to AI, Springer, 2017, Fig. 4.1

Decidability

Theorem

The set of valid formulas in first-order predicate logic is semidecidable.

- A theorem, i.e. a true statement, a prover based on a complete calculus (like resoltion calculus) can verify the truth of the statements.
- For statements that are false, the prover might never halt. Then, it is not possible to decide, whether the prover needs more time to proof a true statement, or whether the statement is false.

Incompleteness I

- Quantifiers over functions or predicates are not included in PL1 (only over variables)
- Sometimes, such an extension might be desirable

Example

Induction:
"If a predicate $p(n)$ holds for n, then $p(n+1)$ also holds"

$$\forall p p(n) \rightarrow p(n+1)$$

However, such extensions will come for the price of loosing decidability.

Incompleteness II

Theorem (Gödel's incompleteness theorem)

Every axiom system for the natural numbers with addition and multiplication (arithmetic) is incomplete. That is, there are true statements in arithmetic that are not provable.

Idea behind Gödel's Theorem:

- Gödelization: every arithmetic formula is encoded as a unique number
- The proposition

  $$F = \text{“\textit{I am not provable.}”}$$

  is formulated in the language of arithmetic.
- This formula is true for the following reason:
  - Assume F is false.
  - Then we can prove F and therefore show that F is not provable. This is a contradiction.
  - Thus F is true and therefore not provable.

Incompleteness III

Practical significance:

- Mathematical theories (axiom systems) become incomplete if the language becomes too powerful.
- Example: Set theory and Russel's paradox.
- Still, undecidable axiom systems might be useful.

Monoticity of Logic I

Example (Flying penguin)

The following statements

1. Tweety is a penguin.
2. Penguins are birds.
3. Birds can fly

can be formalized in PL1 as a knowledge base (forall quantifier suppressed)

$$\begin{aligned} &\mathit{Penguin}(\mathit{tweety}) \\ &\mathit{Penguin}(X) \Rightarrow \mathit{Bird}(X) \\ &\mathit{Bird}(X) \Rightarrow \mathit{Fly}(X). \end{aligned}$$

one can derive (e.g. with resolution) $\mathit{Fly}(\mathit{tweety})$.

Monoticity of Logic II

Example (Flying penguin)

This shows, that we need to introduce an exception for pinguins. We try

$$\mathit{Penguin}(X) \Rightarrow \neg \mathit{Fly}(X)$$

From this, we can derve $\neg \mathit{Fly}(\mathit{tweety})$. But, the **knowledge base is inconsistent**, because $\mathit{Fly}(\mathit{tweety})$ can still be derived.

Monoticity of Logic III

Theorem (Monotonic logic)

A logic is called **monotonic** *if, for an arbitrary knowledge base $\mathcal{A}$ and an arbitrary formula Φ, the set of formulas derivable from $\mathcal{A}$ is a subset of the formulas derivable from $\mathcal{A} \cup \phi$.*

- If a set of formulas is extended, then, all previously derivable statements can still be proved, and additional statements can potentially also be proved.
- The extension of the knowledge base will never lead to our goal.

Monoticity of Logic IV

Example (Flying penguin (contd.))

We try to modify the knowledge base by correcting the obviously false statement *"(all) birds can fly"* by *"(all) birds except penguins can fly"*. The $\mathcal{A}_2$ reads

$$
\begin{aligned}
&\mathit{Penguin}(\mathit{tweety})\\
&\mathit{Penguin}(X) \Rightarrow \mathit{Bird}(X)\\
&\mathit{Bird}(X) \land \neg \mathit{Penguin}(X) \Rightarrow \mathit{Fly}(X)\\
&\mathit{Penguin}(X) \Rightarrow \neg \mathit{Fly}(X)
\end{aligned}
$$

Now, we derive $\neg \mathit{Fly}(\mathit{tweety})$, but not $\mathit{Fly}(\mathit{tweety})$ and $\mathcal{A}_2$ is consistent.

Monoticity of Logic V

Example (Flying penguin (contd.))

Now, we add $\mathit{Raven}(\mathit{abraxas})$

$$\begin{aligned}
&\mathit{Raven}(\mathit{abraxas}) \\
&\mathit{Raven}(X) \Rightarrow \mathit{Bird}(X) \qquad\qquad \mathit{Penguin}(\mathit{tweety}) \\
&\mathit{Penguin}(X) \Rightarrow \mathit{Bird}(X) \\
&\mathit{Bird}(X) \land \neg \mathit{Penguin}(X) \Rightarrow \mathit{Fly}(X) \\
&\mathit{Penguin}(X) \Rightarrow \neg \mathit{Fly}(X)
\end{aligned}$$

We are unable to say anything about the flying capability of Abraxas.

Monoticity of Logic VI

Example (Flying penguin (contd.))

We need to state, that ravens are not penguins:

$$
\begin{aligned}
&\mathit{Raven}(\mathit{abraxas})\\
&\mathit{Raven}(X) \Rightarrow \mathit{Bird}(X)\\
&\mathit{Raven}(X) \Rightarrow \neg \mathit{Penguin}(X)\\
&\mathit{Penguin}(\mathit{tweety})\\
&\mathit{Penguin}(X) \Rightarrow \mathit{Bird}(X)\\
&\mathit{Bird}(X) \wedge \neg \mathit{Penguin}(X) \Rightarrow \mathit{Fly}(X)\\
&\mathit{Penguin}(X) \Rightarrow \neg \mathit{Fly}(X)
\end{aligned}
$$

There are about 10000 different bird species and for each of one, we would have to state, that they are not penguins.

Non-monotonic Logic

- Logic systems, which allows knowledge (formulas) to be removed from the knowledge base, are called **non-monotonous logics**.
- Example: Default logic

Modeling Uncertainty

- Sometimes, a proposition is not just true or false.
- There are different approaches to incorporate uncertainty
  - Fuzzy logic and fuzzy sets
  - Probabilistic logic: We want to assign a probability to a certain statement to express uncertainty, like $Prob(Bird(X) \Rightarrow Fly(X)) = 0.99$.

Chap. 5 Probability theory and probabilistic logic

Chap. 6 Bayesian Networks

Chap. 7 Further Approaches