Principles of Knowledge Representation and Reasoning

Semantic Networks and Description Logics I: Simple, Strict Inheritance Networks

UNI FREIBURG

Bernhard Nebel, Felix Lindner, and Thorsten Engesser

November 16, 2015

1 Introduction

- Motivation

Terminological reasoning

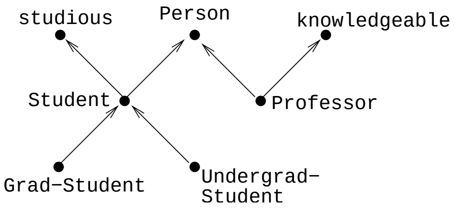
- Often, we need to use semantic (conceptual, terminological) knowledge ...
- For example, consider a knowledge base that classifies things into different categories, which in turn may be organized in some hierarchical way

  Task: Query objects that belong to a specific category or one of its super categories ...
- Even more involved: Answer queries of users of the knowledge base who are not aware of the internal categories of the knowledge base
- Topic of this section: a naïve (maybe too naïve) approach to reasoning with terminological knowledge, namely inheritance networks

Intuition

Definition

A strict inheritance network is defined by a set of nodes (representing concepts, properties) and a set of directed edges (representing generalization, the is-a-relation).



- Reasoning problem: Is some concept C a specialization (a subconcept) of another concept C'?
- . . . and how can we solve this problem efficiently?

2 A simple network formalism

- Semantics
- A polynomial inheritance algorithm
- Soundness & Completeness

UNI FREIBURG

Networks as formula sets

A strict inheritance network can be seen as a set Θ of formulae of the form

$$C_1 \textbf{ isa } C_2.$$

Example

$$\begin{aligned}
\texttt{Student} &\textbf{ isa } \texttt{Person}\\
\texttt{Student} &\textbf{ isa } \texttt{studious}\\
\texttt{Professor} &\textbf{ isa } \texttt{Person}\\
\texttt{Professor} &\textbf{ isa } \texttt{knowledgeable}\\
\texttt{Grad-Student} &\textbf{ isa } \texttt{Student}\\
\texttt{Undergrad-Student} &\textbf{ isa } \texttt{Student}
\end{aligned}$$

Reasoning problem (inheritance problem): $\Theta \models C_1 \textbf{ isa } C_2$?

Logical semantics

- We assign the following logical semantics to **isa**-formulae:

$$C_1 \textbf{ isa } C_2 \quad \mapsto \quad \forall x . C_1(x) \rightarrow C_2(x)$$

- ...i.e., we interpret each directed edge or **isa**-formula as a universally quantified implication.
- This is intuitively plausible: each instance of a sub-concept is an instance of the super-concept.
- Now we can reduce the inheritance problem as follows: Let $\pi(\Theta)$ be the translation. Then we want to know:

$$\pi(\Theta) \models \forall x . C_1(x) \rightarrow C_2(x)\,?$$

- How hard is this problem?

A polynomial reasoning algorithm

Let G_Θ be the graph corresponding to Θ. Then we have:

$$\pi(\Theta) \models \forall x. C_1(x) \rightarrow C_2(x)$$

iff

there exists a path in G_Θ from C_1 to C_2.

- ...which has to be proven (next slides).
- Thus, we have reduced reasoning in strict inheritance networks to graph reachability problem, which is solvable in polynomial time.
- *Note*: Reasoning is not simple because we used a graph to represent the knowledge (there are actually very difficult graph problems),
- ... reasoning is simple because the expressiveness compared with first-order logic is very restricted.

Soundness

Theorem (Soundness of inheritance reasoning)

If there exists a path from C_1 to C_2 in G_Θ, then

$$\pi(\Theta) \models \forall x . C_1(x) \rightarrow C_2(x).$$

Proof.

If there is a path, then there exists a chain of implications of the form $\forall x . D_j(x) \rightarrow D_{j+1}(x)$ with $D_0 = C_1$ and $D_n = C_2$.
Since logical implication is transitive, the claim follows trivially. □

Completeness

Theorem (Completeness of inheritance reasoning)

If $\pi(\Theta) \models \forall x. C_1(x) \rightarrow C_2(x)$, then there exists a path from C_1 to C_2 in G_Θ.

Proof.

We prove the contraposition.
Assume that there exists no such path from C_1 to C_2 in G_Θ. We show that $\pi(\Theta) \not\models \forall x. C_1(x) \rightarrow C_2(x)$.
For this define an interpretation on a universe with exactly one element d such that d is in the interpretation of C_1 and in the interpretation of all concepts reachable from C_1 by following directed edges (and not in the interpretation of any other concept).
This interpretation satisfies all formulae in $\pi(\Theta)$.
However, it does not satisfy $\forall x. C_1(x) \rightarrow C_2(x)$.
For this reason, we have $\pi(\Theta) \not\models \forall x. C_1(x) \rightarrow C_2(x)$. □

UNI FREIBURG

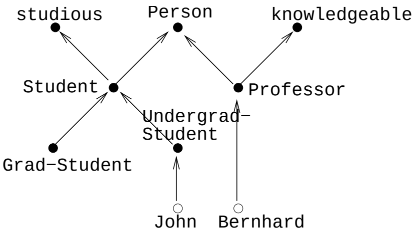
3 Semantic Networks with Instances

UNI FREIBURG

An extension: instances

We also want to talk about instances of concepts.

Example:



… as formulae:

$\vdots$

John **inst-of** Undergrad-Student

Bernhard **inst-of** Professor

Introduction
A simple network formalism
Semantic Networks with Instances
Semantic Networks with Negation
Semantic Networks with Negation and Conjunction
Literature

Extension of the semantics

Logical semantics:

$$i \textbf{ inst-of } C \;\mapsto\; C(i).$$

- Problem 1: Is this extension of the language conservative? That is, can we still decide $\Theta \models C_1 \textbf{ isa } C_2$ without taking formulae of the form i **inst-of** C into account?
- yes (but has to be shown)
- Problem 2: Is it true: $\Theta \models i \textbf{ inst-of } C$ if and only if there is a path from the node i to the node C in G_Θ?
- yes (has to be shown)
- This means, we can also use efficient graph algorithms for this extension.

4 Semantic Networks with Negation

- Satisfiability of a Semantic Network
- Reasoning

A further extension: negated concepts

We now allow for negated concepts, i.e, concept terms of the form

$$\textbf{not}\, C,$$

where C is a concept name (an atomic concept).

Example

$$\texttt{Undergrad-Student}\ \textbf{isa not}\ \texttt{Grad-Student}$$

Logical semantics:

$$\textbf{not}\, C \;\mapsto\; \neg C(x)$$

Example

$$C_1\ \textbf{isa not}\, C_2 \mapsto \forall x. C_1(x) \rightarrow \neg C_2(x).$$

UNI FREIBURG

Complementing an inheritance network

Define $\overline{\alpha}$:

$$\overline{\alpha} := \begin{cases} \textbf{not}\, C & \text{if } \alpha = C \\ C & \text{if } \alpha = \textbf{not}\, C \end{cases}$$

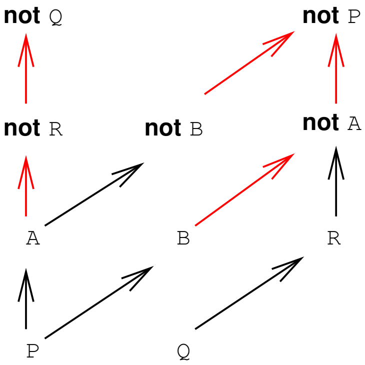
Construct G_Θ from Θ as follows:

- For each concept name C, we will have two nodes: C and **not** C.
- For each formula α_1 **isa** α_2, we introduce the following two edges:

$$\alpha_1 \rightarrow \alpha_2$$
$$\overline{\alpha_2} \rightarrow \overline{\alpha_1}$$

Example

$$\Theta = \{\mathtt{A}\ \textbf{isa not}\ \mathtt{B},\ \mathtt{P}\ \textbf{isa}\ \mathtt{A},\ \mathtt{P}\ \textbf{isa}\ \mathtt{B},\ \mathtt{Q}\ \textbf{isa}\ \mathtt{R},\ \mathtt{R}\ \textbf{isa not}\ \mathtt{A}\}$$



UNI FREIBURG

Satisfiability of an inheritance network

- Strict inheritance networks without negation are always satisfiable, i.e., they have a non-empty model (which one?)
- This is no longer true when we allow for negated concepts. Consider:

  $$\mathtt{P}\ \textbf{isa not}\ \mathtt{P},\ \textbf{not}\ \mathtt{P}\ \textbf{isa}\ \mathtt{P}$$

  means

  $$\forall x.\mathtt{P}(x) \rightarrow \neg\mathtt{P}(x),\ \forall x.\neg\mathtt{P}(x) \rightarrow \mathtt{P}(x),$$

  which is equivalent to

  $$\forall x.\neg\mathtt{P}(x),\ \forall x.\mathtt{P}(x).$$

- … i.e., this set of formulae is not satisfiable, symb. $\Theta \models \bot$.
- This is important to find out since in this case everything follows.

Deciding satisfiability

Theorem (Satisfiability of strict networks with negation)

$\Theta \models \bot$ if and only if the graph G_Θ contains a cycle from α to $\overline{\alpha}$ and back to α.

Proof.

$\Leftarrow$: Adding $\overline{\alpha_2} \rightarrow \overline{\alpha_1}$ corresponds to adding

$$\forall x. \neg\alpha_2(x) \rightarrow \neg\alpha_1(x)$$

when $\forall x. \alpha_1(x) \rightarrow \alpha_2(x)$ is given. This is logically correct (contraposition). Since all directed paths in G_Θ correspond to universally quantified implications that can be deduced from $\pi(\Theta)$, a cycle as in the theorem implies:

$$\forall x. \alpha(x) \rightarrow \overline{\alpha}(x),\ \forall x. \overline{\alpha}(x) \rightarrow \alpha(x).$$

This, however, is unsatisfiable. □

Proof – continued

Proof (cont'd).

$\Rightarrow$: We have to show that unsatisfiability of Θ implies the existence of a cycle from some node α to $\overline{\alpha}$ and back to α in G_Θ.
We prove the contraposition, i.e. that the absence of any such cycle implies satisfiability.
We start with the universe $\mathbf{D} = \{d\}$ and then construct step-wise an interpretation for all concepts.
Convention: Whenever we assign $\alpha^{\mathcal{I}} = \{d\}$, then we assign $\overline{\alpha}^{\mathcal{I}} = \emptyset$.

1. Choose an α without an interpretation that has no path to $\overline{\alpha}$.
2. Assign $\alpha^{\mathcal{I}} = \{d\}$ and continue to do that for all concepts β reachable from α that do not have an interpretation.
3. Continue until all concepts have an interpretation.

If there is still a concept without an interpretation, we always can find one satisfying the condition in step 1 since there is no cycle.
In step 2, no concept reachable from α can have an empty interpretation, so the assignment does not violate any subconcept

isa-Reasoning

Theorem (Inheritance in strict networks with negation)

*$\Theta \models \alpha_1$ **isa** α_2 if and only if one of the following conditions is satisfied:*

1. $\Theta \models \bot$.
2. *There is a path from α_1 to $\overline{\alpha_1}$ in G_Θ.*
3. *There is a path from $\overline{\alpha_2}$ to α_2 in G_Θ.*
4. *There is a path from α_1 to α_2 in G_Θ.*

Proof (sketch).

Soundness is obvious.
Completeness can be shown using the same argument that we used for completeness of the Satisfiability Theorem and the fact that we can start the construction process with $\alpha_1{}^\mathcal{I} = \{d\}$ and $\overline{\alpha_2}^\mathcal{I} = \{d\}$. □

UNI FREIBURG

5 Semantic Networks with Negation and Conjunction

A final extension: conjunctions and negation

A concept description is a concept name (C), a negation of a concept name (**not** C) or the conjunction of concept descriptions (α_1 **and** α_2).

Example

(`Student` **and not** `Grad-Student`) **isa** `Undergrad-Student`

(`Woman` **and** `Parent`) **isa** `Mother`

- Logical semantics is obvious!
- Is it still possible to decide inheritance in polynomial time?

UNI FREIBURG

Computational complexity

Theorem (Complexity of strict inheritance with negation and conjunction)

The reasoning problem for strict inheritance networks with conjunction and negation is coNP-complete.

Proof (sketch).

We show hardness by a reduction from 3SAT.
Let $D = C_1 \wedge \ldots \wedge C_n$ be formula in CNF with exactly three literals per clause (over atoms a_i).
Let $\sigma(C_j)$ be the following translation:

$$a_1 \vee a_2 \vee a_3 \mapsto (\textbf{not}\, a_1 \textbf{ and not}\, a_2) \textbf{ isa } a_3$$

$$\neg a_1 \vee a_2 \vee a_3 \mapsto (a_1 \textbf{ and not}\, a_2) \textbf{ isa } a_3$$

$$\neg a_1 \vee \neg a_2 \vee a_3 \mapsto (a_1 \textbf{ and } a_2) \textbf{ isa } a_3$$

$$\neg a_1 \vee \neg a_2 \vee \neg a_3 \mapsto (a_1 \textbf{ and } a_2) \textbf{ isa } (\textbf{not}\, a_3)$$

Extend σ to CNF formulae, and show that D is unsatisfiable iff $\sigma(D) \models \bot$. □

Conclusion

- Strict inheritance networks are easy
- Inheritance corresponds to a universally quantified implication
- If concepts are atomic, everything can be decided in poly. time
- We can deal with negation without increasing the complexity
- Conjunction and negation, however, make the reasoning problem hard
- . . . as hard as propositional unsatisfiability.

Literature

P. Atzeni, D. S. Parker.
Set Containment Inference and Syllogisms.
Theoretical Computer Science, **62**: 39–65, 1988.