Semantic Networks and Description Logics

Simple, Strict Inheritance Networks

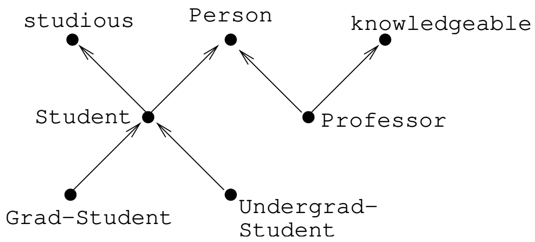
Knowledge Representation and Reasoning

December 5, 2005

Simple, Strict Inheritance Networks – Outline

Intuition

A **strict inheritance network** contains **nodes** (concepts, properties) and **directed edges** (generalization/ISA relation).



- **Reasoning problem**: Is a concept B a **specialization** (a subconcept) of another concept B'?
- **Question**: Can we solve this problem efficiently?

Networks as Formula Sets

A strict inheritance network is a set Θ of formulas of the form

$$C_1 \;\mathbf{isa}\; C_2.$$

Example:

Student	**isa**	Person
Student	**isa**	studious
Professor	**isa**	Person
Professor	**isa**	knowledgeable
Grad-Student	**isa**	Student
Undergrad-Student	**isa**	Student

Reasoning Problem (Inheritance): $\Theta \models C_1 \;\mathbf{isa}\; C_2$.

Logical Semantics

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

$$C_1 \,\mathbf{isa}\, C_2 \;\mapsto\; \forall x\colon\; C_1(x) \rightarrow C_2(x).$$

- We interpret each directed edge or each **isa**-formula as a universally quantified implication
- Conforms with intuition: 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:

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

- How hard is this problem?

A Polynomial Reasoning Algorithm

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

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

iff

there exists a path in G_Θ from C_1 to C_2.

- ...which has to be proven
- We have reduced reasoning in strict inheritance networks to graph reachability problem, which is solvable in poly. 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)

(Soundness) If there is a path from C_1 to C_2 in G_Θ then $\pi(\Theta) \models \forall x\colon\; C_1(x) \rightarrow C_2(x)$.

Proof.

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

Completeness

Theorem (Completeness of inheritance reasoning)

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

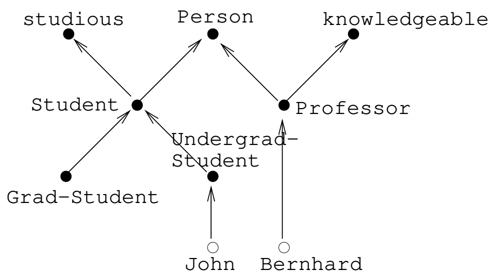
Proof.

We prove the contraposition by constructing a counter example.Suppose the universe has exactly one element d, which is in the interpretation of C_1 and in the interpretation of all concepts reachable from C_1 by following the directed edges. This interpretation satisfies all formulas in $\pi(\Theta)$.However, it does not satisfy $\forall x\colon\; C_1(x) \rightarrow C_2(x)$. For this reason, we have $\pi(\Theta) \not\models \forall x\colon\; C_1(x) \rightarrow C_2(x)$. □

An Extension: Instances

We want to talk about **instances** of concepts.

Example:



As formulas:

$$\vdots$$

`John`	**inst-of**	`Undergrad-Student`
`Bernhard`	**inst-of**	`Professor`

Extension of the Semantics

Logical Semantics

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

- **1st Problem**: Is our extension **conservative**? I.e., can we decide $\Theta \models C_1\ \mathbf{isa}\ C_2$ without taking the formulas $i\ \textbf{inst-of}\ C$ into account?
- yes (has to be shown)
- **2nd Problem**: Is it true: $\Theta \models i\ \textbf{inst-of}\ C$ iff there is a path from i to C in G_Θ?
- yes (has to be shown)
- I.e., we can use our efficient algorithm for this extension.

Another Extension: Negated Concepts

We now allow at all places where we had a concept before the expression

$$\mathbf{not}\, C,$$

where C is a concept.

Example:

$$\texttt{Undergrad-Student} \quad \mathbf{isa} \quad (\mathbf{not}\, \texttt{Grad-Student})$$

Logical semantics:

$$(\mathbf{not}\, C) \mapsto \neg C(x).$$

Example:

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

Completing an Inheritance Network

Define $\overline{\alpha}$:

$$\begin{aligned} \overline{C} &= \mathbf{not}\, C \\ \overline{(\mathbf{not}\, C)} &= C \end{aligned}$$

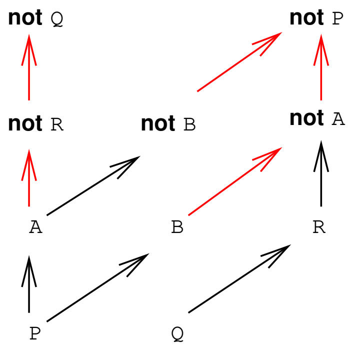
Construct G_Θ from Θ as follows

- For each concept name C, we will have two **nodes**: C and $\mathbf{not}\, C$
- For each formula $\alpha_1\ \mathbf{isa}\ \alpha_2$, we introduce the following two edges:

$$\begin{aligned} \alpha_1 &\longrightarrow \alpha_2 \\ \overline{\alpha_2} &\longrightarrow \overline{\alpha_1} \end{aligned}$$

Example

$$\Theta = \{\ \texttt{A}\ \textbf{isa}\ (\textbf{not}\ \texttt{B}), \texttt{P}\ \textbf{isa}\ \texttt{A}, \texttt{P}\ \textbf{isa}\ \texttt{B}, \texttt{Q}\ \textbf{isa}\ \texttt{R}, \texttt{R}\ \textbf{isa}\ (\textbf{not}\ \texttt{A})\ \}$$



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 not true any longer:

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

means

$$\forall x\colon \texttt{P}(x) \rightarrow \neg\texttt{P}(x), \forall x\colon \neg\texttt{P}(x) \rightarrow \texttt{P}(x),$$

which is equivalent to

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

- The set of formulas is not satisfiable, symbolically $\Theta \models$.
- This is important to find out since in this case everything follows.

Deciding Satisfiability

Theorem (Satisfiability of strict networks with negation)

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

Proof.

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

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

when $\forall x\colon \alpha_1(x) \rightarrow \alpha_2(x)$ 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:

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

This, however, is unsatisfiable. □

Proof – continued

Proof – continued.

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

1. Choose an α without an interpretation, which does not have a path to $\overline{\alpha}$.
2. Assign $\alpha^I = \{d\}$ and continue to do that for all concepts β reachable from α which 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 above α can have an empty interpretation, so the assignment does not violate any subconcept relations.

$\rightsquigarrow$ When the assignment process finishes, we have a model! □

isa Reasoning

Theorem (Inheritance in strict networks with negation)

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

1. $\Theta \models$.
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 $\alpha_1{}^I = \{d\}$ and $\overline{\alpha_2}^I = \{d\}$. □

$\rightsquigarrow$ What about instance-relationship reasoning?

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:

$$(\texttt{Student}\ \textbf{and not}\ \texttt{Grad-Student})\quad \textbf{isa}\quad \texttt{Undergrad-Student}$$
$$(\texttt{Woman}\ \textbf{and}\ \texttt{Parent})\quad \textbf{isa}\quad \texttt{Mother}$$

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

Computational Complexity

Theorem (Complexity of strict inheritance with negation and conjunction)

The reasoning problem for strict inheritance networks with conjunction and negation is co-NP-hard.

Proof.

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:

$$\begin{array}{rcl}
a_1 \vee a_2 \vee a_3 & \mapsto & (\textbf{not}\, a_1\ \textbf{and not}\, a_2)\ \textbf{isa}\ a_3 \\
\neg a_1 \vee a_2 \vee a_3 & \mapsto & (a_1\ \textbf{and not}\, a_2)\ \textbf{isa}\ a_3 \\
\neg a_1 \vee \neg a_2 \vee a_3 & \mapsto & (a_1\ \textbf{and}\ a_2)\ \textbf{isa}\ a_3 \\
\neg a_1 \vee \neg a_2 \vee \neg a_3 & \mapsto & (a_1\ \textbf{and}\ a_2)\ \textbf{isa}\ (\textbf{not}\, a_3)
\end{array}$$

Extend σ to CNF formulas.
Now it is easy to see that D is unsatisfiable iff $\sigma(D) \models$. □

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.