Knowledge Engineering

Semester 2, 2004-05

Michael Rovatsos
mrovatso@inf.ed.ac.uk

Lecture 17 – Knowledge Evolution I: TMS & EBL
11th March 2005

Where are we?

In the last few lectures . . .

- Knowledge Synthesis
- Automated Software Synthesis
- Agents & Multiagent Systems
- Semantic Web & Knowledge Engineering

In the final two lectures . . .

- Knowledge Evolution
- Today:
  - Belief Revision: Truth Maintenance Systems
  - Knowledge in Learning: Explanation-Based Learning

Knowledge Evolution

- So far, we discussed knowledge acquisition, representation & reasoning, and synthesis as if we are always building systems from scratch
- In real-world applications, we expect our KBS to operate over an extended period of time in an environment that changes
  - How to deal with a changing world *considering* our current knowledge?
- Knowledge evolution denotes in this sense the evolution of existing knowledge in the light of new information
- Also an issue for human involvement in the design and implementation of KBS, we will focus on computational aspects
  - Today: belief revision & learning with prior knowledge

Truth Maintenance Systems (TMS)

- In section on non-monotonic reasoning, we mentioned that some inferences have only default status until more specific information is known
- More general problem: **belief revision**, i.e. if we add $\neg P$ to a KB that contains P, how do we make sure all inferences drawn from P are retracted?
  - If $P \Rightarrow Q$, we have to retract Q as well ...
  - but what if also $R \Rightarrow Q$?
- **Truth maintenance systems** (TMS) deal with this problem
- Naive approach:
  - Number all facts P_1 to P_n in the order in which they were added to the KB
  - If P_i is removed, go back to state before addition of P_i and add P_{i+1} to P_n (and what was inferred from them) again
  - Simple, but impractical!

Justification-Based TMS (JTMS)

- Based on idea of annotating each fact with its "justification" (set of logical sentences from which it was inferred)
- Example: A forward-chaining KBS that ads sentences it can infer from existing ones automatically
  - Using JTMS, it will add Q to the KB because of P and $P \Rightarrow Q$ and annotate it with $\{P, P \Rightarrow Q\}$
- A sentence can have several justifications
- If P is to be retracted from the KB, all sentences that require P in *every* justification have to be removed, too
- In the above example: Consider the following justification sets for Q
  - $\{\{P, P \Rightarrow Q\}, \{P, R \vee P \Rightarrow Q\}\}$ ➡ Q will have to be removed
  - $\{\{P, P \Rightarrow Q\}, \{R, R \vee P \Rightarrow Q\}\}$ ➡ Q can be retained

Justification-Based TMS (JTMS)

- Obvious advantage: when retracting P, only those sentences derived from P have to be considered (not all those inferred since P had been added)
- JTMS mark sentences as **in** or **out** (rather than deleting them completely)
  - All inference chains are retained, useful if some facts might become true again
  - Of course, in practice sentences will be eventually deleted if never used again
- Additional advantage (apart from efficient retraction): speed up of analysis of multiple hypothetical situations

Example

- Consider exam schedule with exam e taking place in time-slot t denoted by $Time(e) = t$
  - Concrete schedule: a conjunction $Time(KM) = 6 \wedge Time(KE) = 2 \wedge \ldots Time(PMR) = 12$
  - $Takes(s, e)$ denotes that a student s has to take exam e
- Rule for exam clashes:

$$\exists s\, Takes(s, e) \wedge Takes(s, f) \wedge Time(e) = Time(f) \Rightarrow Clash(e, f)$$

- Consider $Clash(KE, KM)$ with the following justification

$$\{Takes(Moe, KM), Takes(Moe, KE), Time(KM) = 2, Time(KE) = 2,$$
$$Takes(Moe, KE) \wedge Takes(Moe, KM) \wedge Time(KE) = Time(KM) \Rightarrow Clash(KM, KE)\}$$

- Easy to check alternative schedules, e.g. by retracting $Time(KE) = 2$ and asserting $Time(KE) = 5$ (other clashes become immediately visible)

Assumption-Based TMS (ATMS)

- In a JTMS, only one state of the world is represented at a time
- Idea of ATMS: label each sentence with a set of **assumption sets** that would make it true ➡ sentence holds if all assumptions in one of the assumption sets hold
- Way of providing explanations, which may also include assumptions (including contradictory ones)
- Idea: tag sentence "false" with all sets of contradictory assumptions
- ATMS does not strive to reach a state of mutually consistent assumptions, all possibilities are kept in parallel (no backtracking necessary)

Example

- Suppose we have assumptions a_1 to a_5 and sentences A and B with the following assumption sets:
  - A: $\{\{a_1, a_2\}, \{a_2, a_5\}\}$
  - B: $\{\{a_1\}, \{a_2, a_3\}, \{a_4\}\}$
- "*false*: $\{\{a_4, a_5\}\}$" indicates that a_4 and a_5 contradict each other
- Assume we are adding new sentence $A \wedge B \Rightarrow C$, what is the correct set of assumptions?

Example

1. Create cross-product (all pairwise combinations) of assumption sets of A and B:
   $\{\{a_1, a_2\}, \{a_1, a_2, a_3\}, \{a_1, a_2, a_4\}, \{a_1, a_2, a_5\}, \{a_2, a_3, a_5\}, \{a_2, a_4, a_5\}\}$
2. Remove those the contain superfluous assumptions:
   $\{\{a_1, a_2\}, \{a_2, a_3, a_5\}\}$
3. If a label exists for C already, take union of the two labels and delete redundant assumptions (no contradiction testing necessary)
4. If label for C changed, propagate changes to those sentences whose labels depend on C
5. If all labels of C contain contradictions, add these to the label of "false" (and delete those members or supersets thereof from all other nodes)

Knowledge in Learning – EBL

- In our account of inductive learning (decision trees, version spaces) we didn't make use of prior knowledge
- Basic advantage of using prior knowledge: narrowing down the hypothesis space
  - Entailment constraint of pure inductive learning:

$$Hypothesis \wedge Descriptions \models Classification$$

  - Entailment constraint with background knowledge in **explanation-based learning** (EBL):

$$Hypothesis \wedge Descriptions \models Classification$$
$$Background \models Hypothesis$$

- Agent could have derived hypothesis from background knowledge (instance does not add anything factually new)
- However, EBL is a useful method to derive special-purpose knowledge from first-principle theories

Explanation-Based Learning

- Intuition: Explaining why something is a good idea is much easier than coming up with the idea in the first place
- Two-step process:
  1. Construct an explanation of the observation using prior knowledge
  2. Establish a definition of the class of cases for which explanation can be used
- Crucial step: to identify the necessary condition for the steps used in explanation to apply to another case

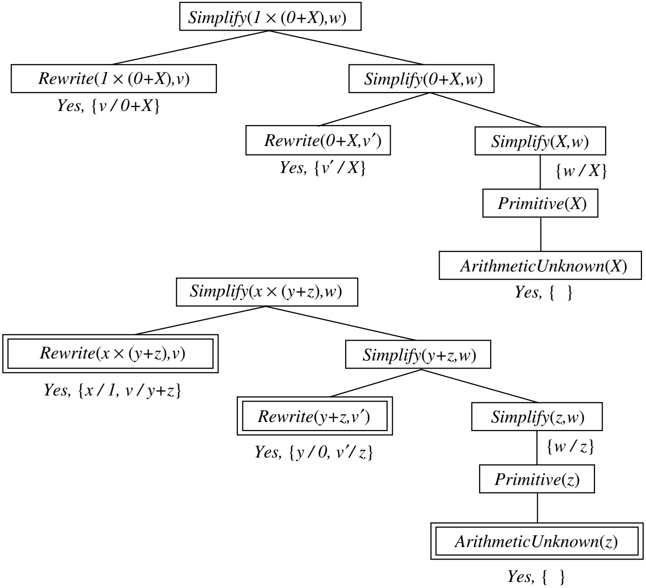
Example

- Suppose we want to simplify the arithmetic expression $1 \times (0 + X)$
- The following set of rules is given for a backward-chaining reasoner:

$$Rewrite(u, v) \wedge Simplify(v, w) \Rightarrow Simplify(u, w)$$
$$Primitive(u) \Rightarrow Simplify(u, u)$$
$$ArithmeticUnknown(u) \Rightarrow Primitive(u)$$
$$Number(u) \Rightarrow Primitive(u)$$
$$Rewrite(1 \times u, u) \quad Rewrite(0 + u, u)$$

- Construct two proof trees in parallel, one with all constants replaced by variables

Example


Simplify(1 × (0+X),w)
Rewrite(1 × (0+X),v)
Yes, {v / 0+X}
Simplify(0+X,w)
Rewrite(0+X,v′)
Yes, {v′ / X}
Simplify(X,w)
{w / X}
Primitive(X)
ArithmeticUnknown(X)
Yes, { }
Simplify(x × (y+z),w)
Rewrite(x × (y+z),v)
Yes, {x / 1, v / y+z}
Simplify(y+z,w)
Rewrite(y+z,v′)
Yes, {y / 0, v′/ z}
Simplify(z,w)
{w / z}
Primitive(z)
ArithmeticUnknown(z)
Yes, { }

Example

- Collect leaf nodes from generalised proof tree to construct a rule for the goal predicate:

$$\begin{aligned}&\mathit{Rewrite}(1 \times (0+z), 0+z) \wedge \mathit{Rewrite}(0+z, z) \wedge \mathit{ArithmeticUnknown}(z)\\ &\qquad \Rightarrow \mathit{Simplify}(1 \times (0+z), z)\end{aligned}$$

- First two conditions don't depend on value of z, this yields simpler rule

$$\mathit{ArithmeticUnknown}(z) \Rightarrow \mathit{Simplify}(1 \times (0+z), z)$$

- More generally, all conditions can be dropped that don't impose rules on values of variables on the RHS of the rule

EBL – Procedure

1. Construct a proof that the goal predicate applies to the example using available background knowledge.
2. In parellel, construct a generalised proof tree fot variabilised goal using the same inference steps as in 1.
3. Construct a new rule whose LHS consists of the leaves of the proof tree and whose RHS is the variabilised goal (while applying appropriate bindings).
4. Drop any conditions that are true regardless of the values of the variables in the goal.

Critique

- As mentioned, nothing actually “new” is learned using the new example, the knowledge is merely “re-formulated”
- Trade-off between generality and specificity of rules:
  - The more general, the more applicable will it be to new cases
  - The more specific, the easier it is to apply (and the less often will it be tried out in vain)
- In practice, EBL is about optimising the choice of appropriate rules with experience (keep different ones and decide empirically which have proven most useful)

Summary

- Discussed some methods that can be used for automating knowledge evolution
- Reasoning maintenance systems: revising beliefs efficiently
- Knowledge in learning: using explanations to reduce hypothesis space
- Explanation-based learning: deriving special-purpose knowledge from general principles
- **Next time: More "knowledge in learning"** (case-based reasoning and/or inductive Logic Programming)