Lecture 10: Induction

2-AIN-144/2-IKV-131 Knowledge Reperesentation & Reasoning

Martin Baláž, Martin Homola

Department of Applied Informatics
Faculty of Mathematics, Physics and Informatics
Comenius University in Bratislava

02 May 2013

Inductive reasoning

Definition (Induction)

Let Γ and $\Delta = \Delta^+ \uplus \Delta^-$ be KBs (sets of formulae) in some language $\mathcal{L}$. A set of formulae Φ in $\mathcal{L}$ is an inductive generalization of Δ (with background theory Γ) if:

- $\Gamma \not\models \Delta^+$
- $\Gamma \cup \Delta$ is consistent
- $\Gamma \cup \Phi \models \Delta^+$
- $\Gamma \cup \Phi \cup \Delta^-$ is consistent

Definition (Induction)

Let Γ and $\Delta = \Delta^+ \uplus \Delta^-$ be KBs (sets of formulae) in some language $\mathcal{L}$. A set of formulae Φ in $\mathcal{L}$ is an inductive generalization of Δ (with background theory Γ) if:

- $\Gamma \not\models \Delta^+$
- $\Gamma \cup \Delta$ is consistent
- $\Gamma \cup \Phi \models \Delta^+$
- $\Gamma \cup \Phi \cup \Delta^-$ is consistent

The task is to induce a generalization Φ that allows to derive the positive observations Δ^+ from the background theory. Negative observations must not be derived.

Definition (Induction)

Let Γ and $\Delta = \Delta^+ \uplus \Delta^-$ be KBs (sets of formulae) in some language $\mathcal{L}$. A set of formulae Φ in $\mathcal{L}$ is an inductive generalization of Δ (with background theory Γ) if:

- $\Gamma \not\models \Delta^+$
- $\Gamma \cup \Delta$ is consistent
- $\Gamma \cup \Phi \models \Delta^+$
- $\Gamma \cup \Phi \cup \Delta^-$ is consistent

The task is to induce a generalization Φ that allows to derive the positive observations Δ^+ from the background theory. Negative observations must not be derived. Hence $\Gamma \cup \Phi$ must be consistent with Δ^- which contains each negative observation $\neg\phi$.

Example

$$\Delta^+ =$$

$$\begin{gathered} reward(card(4, \spadesuit)) \leftarrow \\ reward(card(7, \spadesuit)) \leftarrow \\ reward(card(2, \clubsuit)) \leftarrow \end{gathered}$$

$$\Delta^- =$$

$$\begin{gathered} \neg reward(card(5, \heartsuit)) \leftarrow \\ \neg reward(card(jack, \clubsuit)) \leftarrow \end{gathered}$$

Example (cont.)

$\Gamma =$

$$\begin{aligned}
num(card(X, Y)) &\leftarrow X \leq 10 \\
fig(card(X, Y)) &\leftarrow X \geq jack \\
black(card(X, Y)) &\leftarrow Y = \spadesuit \vee Y = \clubsuit \\
red(card(X, Y)) &\leftarrow Y = \heartsuit \vee Y = \diamondsuit
\end{aligned}$$

$$\begin{aligned}
2 < 3 &\leftarrow \\
&\dots \\
king < ace &\leftarrow \\
X < Z &\leftarrow X < Y, Y < Z \\
X \leq Y &\leftarrow X < Y \vee X = Y \\
X \geq Y &\leftarrow Y \leq X
\end{aligned}$$

Example (cont.)

Possible generalizations:

$\Phi_1 =$

$$reward(X) \leftarrow num(X), black(X)$$

$\Phi_2 =$

$$reward(card(X, Y)) \leftarrow (X = 4, Y = \spadesuit) \vee (X = 7, Y = \spadesuit) \\ \vee (X = 2, Y = \clubsuit)$$

$\Phi_3 =$

$$reward(card(X, Y)) \leftarrow (X \neq 5 \vee Y \neq \heartsuit) \wedge (X \neq jack \vee Y \neq \clubsuit)$$

(And many others...)

Generic Algorithm

- Often large number of generalizations can be found, not all are optimal
- Generic inductive algorithm searches through the space of possible generalizations and stops when an optimal one is found
- The problem is to define which generalizations are optimal
- Out of all reasonable generalizations we prefer those which are least-general
- Too general (Φ_2) but also unreasonably specific (Φ_3) generalizations are not good

θ-subsumption

Definition (θ-subsumption)

Given two clauses (sets of literals) c_1 and c_2 and a subsumption θ, we say that c_1 θ-subsumes c_2 if $c_1\theta \subseteq c_2$.

θ-subsumption

Definition (θ-subsumption)

Given two clauses (sets of literals) c_1 and c_2 and a subsumption θ, we say that c_1 θ-subsumes c_2 if $c_1\theta \subseteq c_2$.

- We also say that c_1 is induced from c_2 and that it is more general (and contrary c_2 is more specific) of the two clauses. We denote this by $c_1 \leq c_2$.

Definition (θ-subsumption)

Given two clauses (sets of literals) c_1 and c_2 and a subsumption θ, we say that c_1 θ-subsumes c_2 if $c_1\theta \subseteq c_2$.

- We also say that c_1 is induced from c_2 and that it is more general (and contrary c_2 is more specific) of the two clauses. We denote this by $c_1 \leq c_2$.
- θ-subsumption allows us to characterize optimal generalizations a least in limited cases.

θ-subsumption (cont.)

Definition (Generalization)

Given a set of clauses C, a clause c is a generalization of C if $c \leq e$ for all $e \in C$.

Definition (Weakest Generalization)

Given a set of clauses C, a clause c is weakest generalization of C if $e \leq c$ for all other generalizations e of C.

Some auxiliary definitions:

Definition (Comparable literals)

Two literals L_1, L_2 are comparable if they use the same predicate, they are of the same arity, and polarity. That is, if $L_1 = P(u_1, \dots, u_n)$ and $L_2 = P(v_1, \dots, v_n)$ where P is possibly negated.

Definition (Co-occurring terms)

A pair of terms t_1, t_2 is co-occurring in comparable literals $L_1 = P(u_1, \dots, u_n)$ and $L_2 = P(v_1, \dots, v_n)$ if $t_1 = u_i$ and $t_2 = v_i$ for for some $i \in \{1, \dots, n\}$.

Computing weakest generalizations (cont.)

Algorithm for **weakest generalization** of clauses c_1, c_2

1. $c := \emptyset$, $\theta_i := \emptyset$ for $i = 1, 2$
2. for every pair of comparable literals L_1, L_2 of C_1 and C_2 do:
   1. call **literal weakest generalization** on $L_1, L_2, \theta_1, \theta_2$ with result L, θ'_1, θ'_2
   2. $\theta_i := \theta'_i$ for $i = 1, 2$
   3. $c := c \cup \{L\}$
3. return c

Algorithm for **literal weakest generalization** of comparable literals L_1, L_2 using initial substitutions θ_1, θ_2

1. for every pair of terms t_1, t_2 co-occurring in L_1, L_2 do:
    1. if $X/t_1 \in \theta_1$ and $X/t_2 \in \theta_2$ for some variable X, $L_i := L_i\{t_i/X\}$ for $i = 1, 2$
2. for every pair of terms $t_1 \neq t_2$ co-occurring in L_1, L_2 do:
    1. let Y be a new variable w.r.t. θ_1, θ_2
    2. $L_i := L_i\{t_i/Y\}$ for $i = 1, 2$
    3. $\theta_i := \theta_i \cup \{Y/t_i\}$ for $i = 1, 2$
3. return L_1, θ_1, θ_2

Computing weakest generalizations (cont.)

Example. . .
(Please consult your notes)

Computing weakest generalizations (cont.)

The algorithm always produces weakest generalizations which may however contain some redundant literals and can be further reduced:

Definition (Equivalent clauses)

Clauses c, d are equivalent (denoted $c \sim d$) if both $c \leq d$ and $d \leq c$.

Definition (Reduced clause)

Clause c is reduced if for every clause e s.t. $e \subseteq c$ and $e \sim c$ we have $e = c$.

Computing weakest generalizations (cont.)

Algorithm for **reduced clause** of clause c

1. while there is $L \in c$ and a substitution σ s.t. $c\sigma \subseteq c \setminus L$ do:
    1. $c := c\sigma$
2. return c

Final remarks

Theta subsumption allows us to find suitable generalizations in with only positive observations (set of clauses C) with no background knowledge. More general cases are less straight forward for examples of some more general methods see Šefránek (2000) pp. 188–194.

References:

- Šefránek, J.:Inteligencia ako výpočet. IRIS, 2000.