EXERCISE 4

Science of Computational Logic

Steffen Hölldobler, Marcos Cramer

International Masters Programme in Computational Logic — winter semester 2018/2019

17.01.2019

Problem 4.1

Consider the knowledge base

$$\mathcal{F} = \left\{ \begin{array}{rcl} \texttt{interesting-food} & \leftrightarrow & \texttt{dessert} \lor \texttt{spinach-pilaf} \\ \texttt{dessert} & \leftrightarrow & \texttt{magic-cookie-bars} \lor \texttt{banana-burrito} \end{array} \right\}$$

the set of abducibles

$$\mathcal{F}_A = \{ \texttt{ spinach-pilaf, magic-cookie-bars , banana-burrito } \}.$$

and an empty set of integrity constraints. Compute the set of possible explanations for the observation "`interesting-food`"

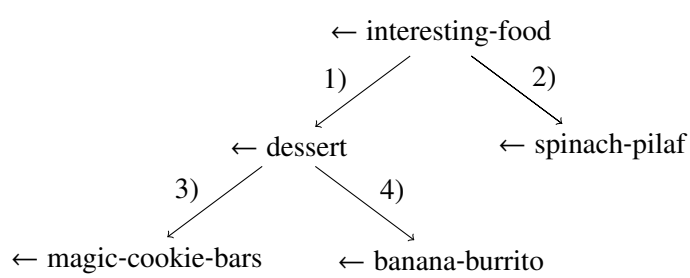
- by using SLD–resolution, and
- by model generation.

Solution

- by SLD-resolution:

$$\begin{array}{rll} \mathcal{F}_{\leftarrow} = \{ & 1) & \text{interesting-food} \leftarrow \text{dessert}, \\ & 2) & \text{interesting-food} \leftarrow \text{spinach-pilaf}, \\ & 3) & \text{dessert} \leftarrow \text{magic-cookie-bars}, \\ & 4) & \text{dessert} \leftarrow \text{banana-burrito} \} \end{array}$$

SLD search space:



Negation of leafs yields possible explanations:

{magic-cookie-bars}

{banana-burrito}

{spinach-pilaf}

- by model generation:

$\mathcal{F}_{\rightarrow} \cup \{$ interesting-food$\} = \{$
1) interesting-food $\rightarrow$ dessert $\vee$ spinach-pilaf,
2) dessert $\rightarrow$ magic-cookie-bars $\vee$ banana-burrito,
3) interesting-food $\}$

The minimal models of this set are:

$$\{\text{interesting-food, spinach-pilaf}\}$$
$$\{\text{interesting-food, dessert, magic-cookie-bars}\}$$
$$\{\text{interesting-food, dessert, banana-burrito}\}$$

Restricting these models to abducible predicates yields the following explanations:

$$\{\text{spinach-pilaf}\}$$
$$\{\text{magic-cookie-bars}\}$$
$$\{\text{banana-burrito}\}$$

Problem 4.2

Specify an abductive framework $\langle \mathcal{F}, \mathcal{F}_A, \mathcal{I} \rangle$ and an observation G, such that the observation can be explained according to the satisfiability view in a way that is not available by the theoremhood view.

Solution

For example:

$$\langle \mathcal{F}, \mathcal{F}_{\mathrm{A}}, \mathcal{I} \rangle \quad \text{with}$$
$$\mathcal{F} = \{b \leftarrow a \vee c\}$$
$$\mathcal{F}_{\mathrm{A}} = \{a, c\}$$
$$\mathcal{I} = \{a \rightarrow c\}$$
$$G = b$$

- In the satisfiability view, $\{a\}$ is an explanation, because

$$\mathcal{F} \cup \{a\} \models b$$
and $\mathcal{F} \cup \{a\} \cup \mathcal{I}$ is satisfiable
(by the interpretation $\{a, b, c\}$)

- In the theoremhood view, $\{a\}$ is not an explanation, because

$$\mathcal{F} \cup \{a\} \not\models \mathcal{I}$$
($\{a, b\}$ is a model for $\mathcal{F} \cup \{a\}$, but no model for $\mathcal{I}$)

Problem 4.3

Assume that you have the data structure `char` of ASCII characters available.

1. Define the data structure `string` according to the following specification:
   A string may be empty or may be obtained by adding an ASCII character to the end of a string.
   Which are the constructors? Which are the selectors?

2. Express explicitly the following conditions that the data structure `string` should satisfy:

   (a) Different constructors produce different objects;

   (b) Constructors of arity > 0 induce injective mappings on the set of constructor ground terms;

   (c) Each constructor ground term can be represented as an application of some constructor to the results of application of selectors, if any applicable selectors exists;

   (d) Each selector is 'inverse' to the constructor it belongs to;

3. Write a program $\mathcal{F}_{Trans}$ that defines the function *Trans* over non-empty strings, which transforms any string into a string of the same length containing only the character 'a'.

Solution

1. $\Lambda \;:\; string$
   $\circ \;:\; string \times char \to string$] constructors

   $h \;:\; string \to string$] selectors (= Initial part)
   $t \;:\; string \to char$ (= tail element)

   ($\circ$ will be written infix)

2. (a) $(\forall S : string,\ C : char)(\Lambda \not\approx S \circ C)$

   (b) $(\forall S, S' : string,\ C, C' : char)(S \circ C \approx S' \circ C' \to (S \approx S' \wedge C \approx C'))$

   (c) $(\forall S : string)(S \approx \Lambda \vee S \approx h(S) \circ t(S))$

   (d) $(\forall S : string,\ C : char)(h(S \circ C) \approx S)$
   $(\forall S : string,\ C : char)(t(S \circ C) \approx C)$

3. $\mathcal{F}_{\text{trans}} = \{\ (\forall X : string)(Trans(X) \approx \Lambda \ \leftarrow\ X \approx \Lambda),$
   $(\forall X : string)(Trans(X) \approx Trans(h(X)) \circ 'a' \ \leftarrow\ X \not\approx \Lambda)\ \}$

   ($\text{Trans}(\Lambda) \approx \Lambda$ is ok, too)