Computational Logic

Extraction of Answers

Damiano Zanardini

UPM European Master in Computational Logic (EMCL)
School of Computer Science
Technical University of Madrid
damiano@fi.upm.es

Academic Year 2009/2010

Introduction

From ATP to extraction of answers

- the techniques for automated theorem proving can be also used for designing systems for extracting answers and solving problems

The idea

- the facts which are needed to find an answer or solve a problem can be seen as axioms or premises
- the question or the problem can be seen as a theorem to be proven

Introduction

Kinds of questions (and answers)

(A) yes/no questions
- is Luís in Madrid? Yes, Luís is in Madrid

(B) questions like *where is*, *who is*, *under which conditions*, ...
- where is Luís? Luís is in Madrid

(C) questions whose answer is a sequence of actions
- what do I have to do? Go to Madrid and take the train

(D) questions whose answer includes verifying some conditions
- what do I have to do? If there are still seats, go to Madrid and take the train, otherwise take the bus

Type A

The correspondence

Since the answer can only be *yes* or *no*, it can be obtained by solving the deduction problem

given $A_1, .., A_n$, is P certainly true? $\rightsquigarrow$ $[A_1, .., A_n] \vdash P$

Example

if one is in Madrid, then (s)he's not in Lugo	$\neg p(x, Madrid) \vee \neg p(x, Lugo)$
Luís is in Madrid	$p(Luis, Madrid)$
is Luís in Lugo?	$\neg p(Luis, Lugo)$

- it's not possible to derive $\Box$ from this, so that we should not answer that Luís is in Lugo
- if the conclusion cannot be proven, then we should try to prove its negation
- if neither can be proven, then the answer should be *not enough information*

Type B

The grandfather

if y is x's father and z is y's father, then z is x's grandfather	C_1
everyone has a father	C_2
who is a's grandfather?	C_0

$C_1 : \neg p(y_1, x_1) \vee \neg p(z_1, y_1) \vee q(z_1, x_1) \qquad C_2 : p(f(x_2), x_2)$

$\{y_1/f(x_2), x_1/x_2\}$

$\neg p(z_1, f(x_2)) \vee q(z_1, x_2) \qquad C_2 : p(f(x_2'), x_2')$

$\{z_1/f(f(x_2)), x_2'/f(x_2)\}$

$q(f(f(x_2)), x_2) \qquad C_0 : \neg q(y_0, a)$

$\alpha = \{y_0/f(f(a)), x_2/a\}$

$\square$

Type B

The grandfather

if y is x's father and z is y's father, then z is x's grandfather	C_1
everyone has a father	C_2
who is a's grandfather?	C_0

$$C_1 : \neg p(y_1, x_1) \vee \neg p(z_1, y_1) \vee q(z_1, x_1) \qquad C_2 : p(f(x_2), x_2)$$

$$\{y_1/f(x_2), x_1/x_2\}$$

$$\neg p(z_1, f(x_2)) \vee q(z_1, x_2) \qquad C_2 : p(f(x_2'), x_2')$$

$$\{z_1/f(f(x_2)), x_2'/f(x_2)\}$$

$$q(f(f(x_2)), x_2) \qquad C_0 : \neg q(y_0, a) \;\vee\; ans(y_0)$$

$$\alpha = \{y_0/f(f(a)), x_2/a\}$$

$$ans(f(f(a)))$$

Type C

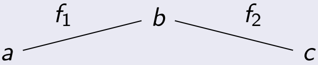
The task

Find a sequence of actions for reaching a goal

- every *object* is supposed to be in a given *state*
- to reach the goal, the state has to be changed to the desired state
- ATP can be used for finding the actions which can produce the change

Type C

Example



- $p(x, y, z)$: x is in state z at y
- $f_1(x, a, b, z)$: final state obtained by moving from a to b the object x which is in state z
- $f_2(x, b, c, z)$: final state obtained by moving from b to c the object x which is in state z

how can d go from a to c?
d is initially in a
with state s_1

$$\begin{array}{ll}
C_0: & \neg p(d, c, z) \vee ans(z) \\
C_1: & p(d, a, s_1) \\
C_2: & \neg p(d, a, z) \vee p(d, b, f_1(d, a, b, z)) \\
C_3: & \neg p(d, b, z) \vee p(d, c, f_2(d, b, c, z))
\end{array}$$

Type C

Example

$p(d, a, s_1)$ $\quad$ $\neg p(d, a, z) \vee p(d, b, f_1(d, a, b, z))$

$p(d, b, f_1(d, a, b, s_1))$ $\quad$ $\neg p(d, b, z) \vee p(d, c, f_2(d, b, c, z))$

$p(d, c, f_2(d, b, c, f_1(d, a, b, s_1)))$ $\quad$ $\neg p(d, c, z) \vee ans(z)$

$ans(f_2(d, b, c, f_1(d, a, b, s_1)))$

- f_1 takes d from a to b
- f_2 takes d from b to c

Type C

The monkey and the banana

Predicates

- $p(x, y, z, s)$: in the state s, the monkey is at x, the banana is at y and the chair is at z
- $r(s)$: in the state s, the monkey can reach the banana

Functions

- $walks(y, z, s)$: the state reached when the monkey walks from y to z starting in the state s
- $takes(y, z, s)$: the state reached when the monkey, starting in the state s, walks from y to z taking the chair with itself
- $climbs(s)$: the state reached when the monkey, starting in the state s, climbs the chair

Type C

The monkey and the banana

Axioms

- $p(a, b, c, s_1)$
- $\neg p(x, y, z, s) \vee p(z, y, z, walks(x, z, s))$
- $\neg p(x, y, x, s) \vee p(y, y, y, takes(x, y, s))$
- $\neg p(x, x, x, s) \vee r(climbs(s))$

Question

- $\neg r(s) \vee ans(s)$

Do these axioms allow the monkey to do whatever it wants?

Type D

The idea

- the task is to find a sequence of actions which, *under certain conditions*, can take to the goal
- it makes sense when the given information does not allow a definite decision

How it works

- every *object* is supposed to be in a given *state*
- to reach the goal, the state has to be changed to the desired state
- ATP can be used for finding the actions which can produce the change, but the application of the actions may be dependent on certain conditions
- the *resolution tree* can be transformed into a *decision tree* by introducing an algorithm for extracting information

Type D

Example

- if someone is younger than 5, then (s)he has to take medicine a

$$C_1 : \ \neg p(x) \vee r(x, a)$$

- if someone is not younger than 5, then (s)he has to take medicine b

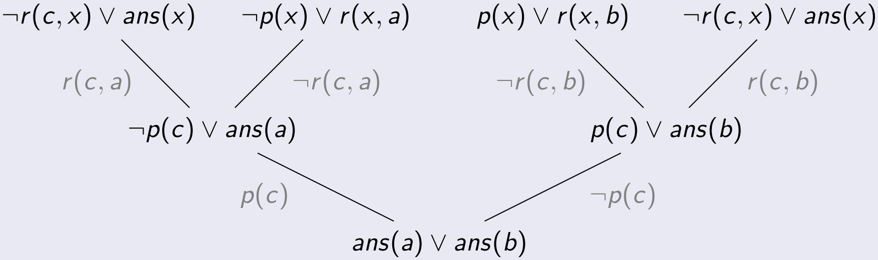
$$C_1 : \ p(x) \vee r(x, b)$$

- which medicine should Carl take?

$$C_0 : \ \neg r(c, x) \vee ans(x)$$

Type D

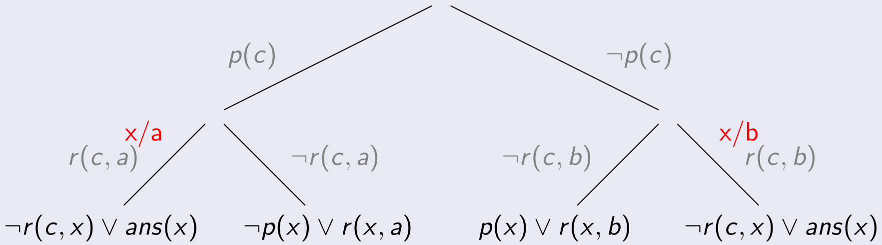
Example



- let $C\alpha \vee D\alpha$ be the resolvent of $L' \vee C$ and $\neg L'' \vee D$, with $\alpha = MGU(L', L'')$
- let e' be the edge from $L' \vee C$ to $C\alpha \vee D\alpha$
- let e'' be the edge from $\neg L'' \vee D$ to $C\alpha \vee D\alpha$
- then, e' is labelled with $\neg L'\alpha$ (note the $\neg$) and e'' is labelled with $L''\alpha$ (note that there is no $\neg$)

Type D

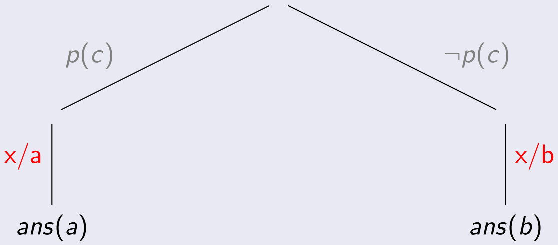
Example



- put the tree upside-down and remove clauses from non-leaf nodes

Type D

Example



- ignore paths leading to clauses without *ans*, and clean irrelevant parts

Conclusion

Completeness

- resolution is complete for answer extraction: if a question has an answer, then an *answer clause* can be deduced by resolution

Questions and answers (type B and C)

- let $\mathcal{C}$ a set of clauses, representing *facts*
- let find values for $x_1..x_k$ such that $p(x_1..x_k)$ holds be the question
- the question has an answer iff $\mathcal{C} \vdash \exists x_1..\exists x_k p(x_1..x_k)$
- the *query* Q will be $\neg p(x_1..x_k) \vee ans(x_1..x_k)$

Theorem

The question has an answer iff there exists a deduction of an answer clause starting from $\mathcal{C} \cup \{Q\}$

- *resolution not only tells if there is an answer, but also what this answer is*