A Reasoner for Simple Conceptual Logic Programs

Stijn Heymans   Cristina Feier   Thomas Eiter

Vienna University of Technology

25 October RR 2009

$$
\begin{aligned}
\mathit{fail}(X) &\leftarrow \mathit{not}\ \mathit{study}(X) \\
\mathit{study}(\mathit{john}) &\leftarrow
\end{aligned}
$$

$$\begin{array}{rcl} \mathit{fail}(X) & \leftarrow & \mathit{not}\ \mathit{study}(X) \\ \mathit{study}(\mathit{john}) & \leftarrow & \end{array}$$

LP	Herbrand Universe $\{\mathit{john}\}$
model	$\{\mathit{study}(\mathit{john})\}$
fail satisfiable?	No

$$
\begin{array}{rcl}
\mathit{fail}(X) & \leftarrow & \mathit{not}\ \mathit{study}(X) \\
\mathit{study}(\mathit{john}) & \leftarrow &
\end{array}
$$

LP	Herbrand Universe $\{\mathit{john}\}$
model	$\{\mathit{study}(\mathit{john})\}$
fail satisfiable?	No

Desirable when Conceptually Modeling?

no

as in FOL or DLs

assume anonymous elements are present

$$
\begin{aligned}
\mathit{fail}(X) &\leftarrow \mathit{not}\ \mathit{study}(X) \\
\mathit{study}(\mathit{john}) &\leftarrow
\end{aligned}
$$

Open ASP	Possible Universe $\{\mathit{john}, x\}$
model	$\{\mathit{study}(\mathit{john}), \mathit{fail}(x)\}$
fail satisfiable?	Yes

$$
\begin{aligned}
\mathit{fail}(X) &\leftarrow \mathit{not}\ \mathit{study}(X) \\
\mathit{study}(\mathit{john}) &\leftarrow
\end{aligned}
$$

Open ASP	Possible Universe $\{\mathit{john}, x\}$
model	$\{\mathit{study}(\mathit{john}), \mathit{fail}(x)\}$
fail satisfiable?	Yes

Conceptual Modeling? Check!

Conceptual Modeling problems solved?

no

satisfiability checking undecidable

syntactically restrict programs:

Simple Conceptual Logic Programs

Simple Conceptual Logic Programs

- only unary and binary predicates

Simple Conceptual Logic Programs

- only unary and binary predicates
- no constants

Simple Conceptual Logic Programs

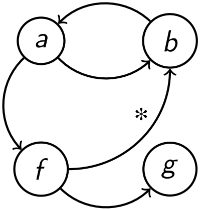
- only unary and binary predicates
- no constants
- 3 rule types (unary, binary, free) with tree shape

Simple Conceptual Logic Programs

- only unary and binary predicates
- no constants
- 3 rule types (unary, binary, free) with tree shape
- cyclicity restriction

$$
\begin{array}{rrcl}
r_1: & a(X) & \leftarrow & b(X), f(X,Y), not\ a(Y) \\
r_2: & b(X) & \leftarrow & a(X) \\
r_3: & f(X,Y) & \leftarrow & g(X,Y), b(Y) \\
r_4: & g(X,Y) \vee not\ g(X,Y) & \leftarrow &
\end{array}
$$

$$
\begin{array}{rrcl}
r_1: & a(X) & \leftarrow & b(X), f(X,Y), \mathit{not}\ a(Y) \\
r_2: & b(X) & \leftarrow & a(X) \\
r_3: & f(X,Y) & \leftarrow & g(X,Y), b(Y) \\
r_4: & g(X,Y) \vee \mathit{not}\ g(X,Y) & \leftarrow &
\end{array}
$$



Satisfiability checking decidable

Motivation from the Hybrid knowledge viewpoint:

let's look at *DL-safeness*

$$\mathit{Student} \quad \sqsubseteq \quad \mathit{Person}$$

$$\mathit{works}(X) \quad \leftarrow \quad \mathit{not}\ \mathit{Student}(X)$$

rule is not DL-safe:

$$\mathit{works}(X) \quad \leftarrow \quad \mathit{not}\ \mathit{Student}(X)$$

$\mathit{Student}$ is a DL-atom, thus X is not *guarded* by positive non-DL atom what can one do?

not a lot

$$\mathit{Student} \quad \sqsubseteq \quad \mathit{Person}$$

$$\mathit{works}(X) \quad \leftarrow \quad \mathbf{Person}(\mathbf{X}), \mathit{not}\ \mathit{Student}(X)$$

still not DL-safe
($\mathsf{Person}(X)$ is also a DL-atom)

Why DL-safeness?

Herbrand

avoid Herbrand and DL-safe

ai, undecidable again

$\Rightarrow$ simple Conceptual Logic Programs

$\Rightarrow$ simple Conceptual Logic Programs

- decidable (in combination with a translatable DL)

⇒ simple Conceptual Logic Programs

- decidable (in combination with a translatable DL)
- not Herbrand

$\Rightarrow$ simple Conceptual Logic Programs

- decidable (in combination with a translatable DL)
- not Herbrand
- not DL-safe

s/DL-safe rules/simple Conceptual Logic Programs

$$\mathit{Student} \quad \sqsubseteq \quad \mathit{Person}$$

$$\mathit{works}(X) \quad \leftarrow \quad \mathit{not}\ \mathit{Student}(X)$$

$$\mathit{Student} \quad \sqsubseteq \quad \mathit{Person}$$

$$\mathit{works}(X) \quad \leftarrow \quad \mathit{not}\ \mathit{Student}(X)$$

- not DL-safe

$$\mathit{Student} \sqsubseteq \mathit{Person}$$

$$\mathit{works}(X) \leftarrow \mathit{not}\ \mathit{Student}(X)$$

- not DL-safe
- simple Conceptual Logic Program

Decidable is nice, but what about actual reasoning?

this paper

- Tableau algorithm for Simple Conceptual Logic Programs

- Tableau algorithm for Simple Conceptual Logic Programs
- Prototype implementation in BProlog

Why is this hard?

LP	closed domain	**minimal model**
DL	**open domain**	model

simple CoLPs: open domain, minimal model

Tableau algorithm for simple CoLPs

- DL-like Tableaux algorithm: build tree structure, blocking

Tableau algorithm for simple CoLPs

- DL-like Tableaux algorithm: build tree structure, blocking
- Extra: Dependency graph for minimality checks

check satisfiability of a:

$$
\begin{array}{llll}
r_1: & a(X) & \leftarrow & f(X, Y_1), b(Y_1), \mathit{not}\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2: & b(X) & \leftarrow & f(X, Y), \mathit{not}\ c(Y) \\
r_3: & c(X) & \leftarrow & \mathit{not}\ b(X)
\end{array}
$$

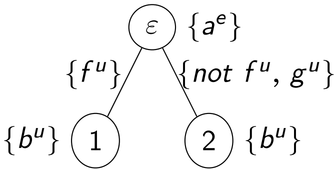
with f and g free, i.e., with rules

$$
\begin{array}{rl}
f(X, Y) \vee \mathit{not}\ f(X, Y) & \leftarrow \\
g(X, Y) \vee \mathit{not}\ g(X, Y) & \leftarrow
\end{array}
$$

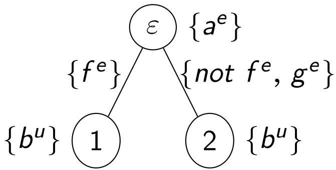
$\varepsilon \quad \{a^u\}$

dependency graph G contains the corresponding atom $a(\varepsilon)$

$$
\begin{array}{rcll}
r_1: & a(X) & \leftarrow & f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2: & b(X) & \leftarrow & f(X, Y), not\ c(Y) \\
r_3: & c(X) & \leftarrow & not\ b(X)
\end{array}
$$



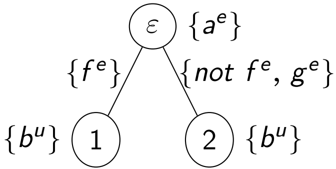
$$
\begin{array}{llcl}
r_1: & a(X) & \leftarrow & f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2: & b(X) & \leftarrow & f(X, Y), not\ c(Y) \\
r_3: & c(X) & \leftarrow & not\ b(X)
\end{array}
$$



- Expand only nodes if ancestors have been fully expanded

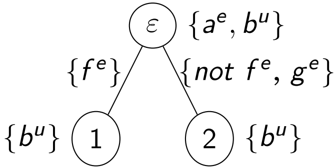
- Expand only nodes if ancestors have been fully expanded
- Each label contains each unary (binary) predicate or its negation (a *saturation*)

$$
\begin{array}{llll}
r_1: & a(X) & \leftarrow & f(X,Y_1), b(Y_1), \mathit{not}\ f(X,Y_2), g(X,Y_2), b(Y_2) \\
r_2: & b(X) & \leftarrow & f(X,Y), \mathit{not}\ c(Y) \\
r_3: & c(X) & \leftarrow & \mathit{not}\ b(X)
\end{array}
$$

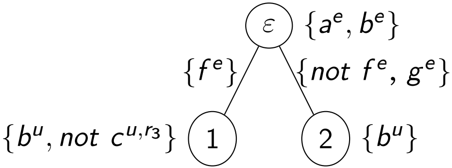


$\Rightarrow$ Root node not saturated yet

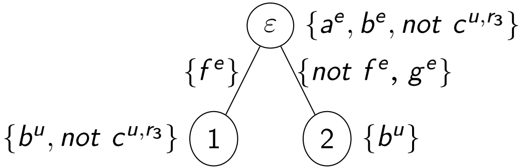
$$
\begin{array}{rcl}
r_1: \quad a(X) & \leftarrow & f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2: \quad b(X) & \leftarrow & f(X, Y), not\ c(Y) \\
r_3: \quad c(X) & \leftarrow & not\ b(X)
\end{array}
$$



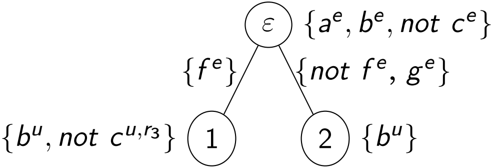
$$
\begin{aligned}
r_1 : \quad a(X) &\leftarrow f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2 : \quad b(X) &\leftarrow f(X, Y), not\ c(Y) \\
r_3 : \quad c(X) &\leftarrow not\ b(X)
\end{aligned}
$$



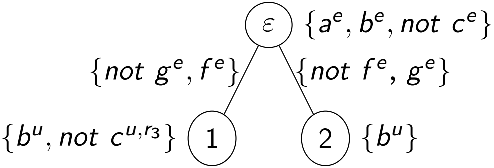
$$
\begin{array}{lrcl}
r_1: & a(X) & \leftarrow & f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2: & b(X) & \leftarrow & f(X, Y), not\ c(Y) \\
r_3: & c(X) & \leftarrow & not\ b(X)
\end{array}
$$



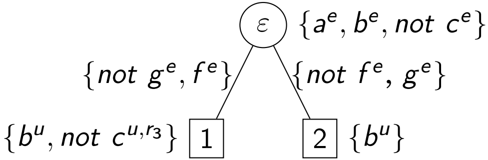
$$
\begin{aligned}
r_1 : \quad a(X) &\leftarrow f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2 : \quad b(X) &\leftarrow f(X, Y), not\ c(Y) \\
r_3 : \quad c(X) &\leftarrow not\ b(X)
\end{aligned}
$$



$$
\begin{aligned}
r_1 : \quad a(X) &\leftarrow f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\
r_2 : \quad b(X) &\leftarrow f(X, Y), not\ c(Y) \\
r_3 : \quad c(X) &\leftarrow not\ b(X)
\end{aligned}
$$

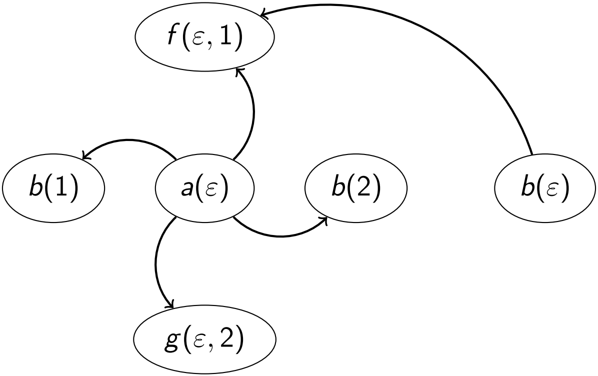


$$
\begin{aligned}
r_1 :\quad a(X) &\leftarrow f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2)\\
r_2 :\quad b(X) &\leftarrow f(X, Y), not\ c(Y)\\
r_3 :\quad c(X) &\leftarrow not\ b(X)
\end{aligned}
$$



$$\begin{array}{llll} r_1: & a(X) & \leftarrow & f(X, Y_1), b(Y_1), not\ f(X, Y_2), g(X, Y_2), b(Y_2) \\ r_2: & b(X) & \leftarrow & f(X, Y), not\ c(Y) \\ r_3: & c(X) & \leftarrow & not\ b(X) \end{array}$$

cycle-free dependency graph:



a is satisfiable

- sound, complete, and terminating

- sound, complete, and terminating
- EXPTIME

BProlog implementation

http://www.kr.tuwien.ac.at/staff/heymans/priv/oasp-r/