

# Open Answer Set Programming: an Overview

Talk at Free University of Bolzano

Stijn Heymans

29 January 2008

# Overview

Problem: Closed-world Reasoning in Answer Set Programming hampers Conceptual Reasoning

Solution: Open Answer Set Programming

Problem with Solution: Undecidable

Solution for Problem with Solution: Identify Decidable Fragments

## Closed-World Reasoning in Answer Set Programming

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

→ *ground* the program with all constants (*john*):

$$\begin{aligned}fail(john) &\leftarrow not\ pass(john) \\ pass(john) &\leftarrow\end{aligned}$$

→ answer set:  $\{pass(john)\}$ .

## Closed-World Reasoning in Answer Set Programming (2)

- ▶ Answer set:  $\{pass(john)\}$ .
- ▶ No *fail*-atom: the *fail*-predicate is not satisfiable.
- ▶ In the context of conceptual reasoning this is not feasible: other *data* make *fail* satisfiable, i.e., the program *makes sense* but one is forced to introduce all significant constants.
- ▶ Do not assume all possible constants are present: assume the presence of anonymous objects – *open domains*.

# Open Answer Set Programming

An *open answer set* of  $P$  is a pair  $(U, M)$  where

- ▶ the *universe*  $U$  is a non-empty superset of the constants in  $P$ , and
- ▶  $M$  is an answer set of  $P_U$ .

Examples:

- ▶  $(\{john, x\}, \{pass(john), fail(x)\})$  is open answer set since  $\{pass(john), fail(x)\}$  is an answer set of

$$\begin{array}{ll} fail(x) & \leftarrow not\ pass(x) \\ fail(john) & \leftarrow not\ pass(john) \\ pass(john) & \leftarrow \end{array}$$

- ▶  $(\{john, x_1, x_2, \dots\}, \{pass(john), fail(x_1), fail(x_2), \dots\})$ ,
- ▶  $(\{john\}, \{pass(john)\})$ .

# Undecidability of Open ASP

→ show by reduction from undecidable *domino problem*.

## Theorem

*Let  $\mathcal{D}$  be a domino system and  $d$  a domino in  $\mathcal{D}$ . Then,  $\mathcal{D}$  tiles the plane  $\mathbb{N} \times \mathbb{N}$  such that  $d$  is present in the tiling iff  $d$  is satisfiable w.r.t.  $[\mathcal{D}]$ .*

# Regaining Decidability

Retain openness, but restrict shape logic programs in order to obtain decidability.

Three types of restrictions:

- ▶ Conceptual Logic Programs
- ▶ Local Forest Logic Programs (and variations)
- ▶ Guarded Programs (and variations)

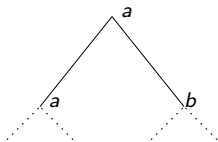
# Conceptual Logic Programs

Reduction of decidability of satisfiability checking to checking non-emptiness of two-way alternating tree automata (2ATA).

## 2ATA

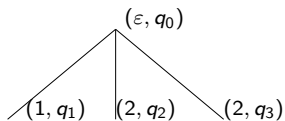
- ▶ Tuple  $(\Sigma, Q, q_0, \delta, \Omega)$  with
  - ▶ alphabet  $\Sigma$ ,
  - ▶ states  $Q$ ,
  - ▶ begin state  $q_0$ ,
  - ▶ transition function  $\delta$ ,
  - ▶ acceptance condition  $\Omega$ .
- ▶ Input to a 2ATA are labeled infinite trees.
- ▶ A run over  $t : T \rightarrow \Sigma$  is a tree  $r : R \rightarrow T \times Q$ .

## 2ATA: Example

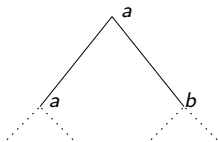


Transition:

$$\delta(q_0, a) = \delta(q_1, a) = (-1, q_1) \vee ((1, q_1) \wedge (2, q_2) \wedge (2, q_3)).$$

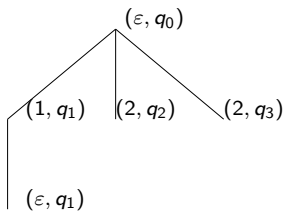


## 2ATA: Example

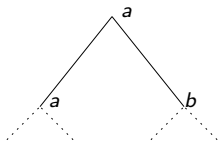


Transition:

$$\delta(q_0, a) = \delta(q_1, a) = (-1, q_1) \vee ((1, q_1) \wedge (2, q_2) \wedge (2, q_3)).$$

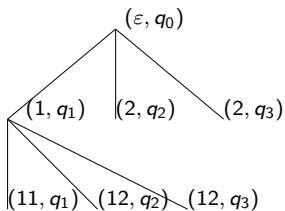


## 2ATA: Example



Transition:

$$\delta(q_0, a) = \delta(q_1, a) = (-1, q_1) \vee ((1, q_1) \wedge (2, q_2) \wedge (2, q_3)).$$



## 2ATA

Assume the automaton may not visit  $q_1$  infinitely: acceptance condition  $(\{q_1\}, Q)$ .

### Acceptance

A run is accepting if for every infinite path  $\pi$  in the run:  
 $inf(\pi) \cap \{q_1\} = \emptyset$  and  $inf(\pi) \cap Q \neq \emptyset$ .

### Non-emptiness

Non-emptiness checking of a 2ATA is checking whether there are infinite trees that are accepted by the 2ATA.

### Complexity

Non-emptiness checking is in EXPTIME w.r.t. the number of states in the 2ATA.

## From OASP to 2ATA

- ▶ Given a program  $P$  and a predicate  $p$ , construct a 2ATA  $A_{p,P}$  such that  $p$  is satisfiable w.r.t.  $P$  iff  $A_{p,P}$  is non-empty.
- ▶ Rewrite open answer set as tree, feed the tree to  $A_{p,P}$  and show that it accepts the tree.
- ▶ Vice versa, take accepted tree of  $A_{p,P}$  and show that it can be written as open answer set that satisfies  $p$ .

→ Search for programs with tree-model property (*open answer sets can be rewritten as trees*).

# Conceptual Logic Programs

- ▶ Only unary and binary predicates allowed:  $a(X)$  and  $f(X, Y)$ .
- ▶ No constants.
- ▶ Four types of rules:
  - ▶ Free rules
  - ▶ Unary rules
  - ▶ Binary rules
  - ▶ Constraints

## Free Rules

$a(X) \vee \text{not } a(X) \leftarrow$  or  $f(X, Y) \vee \text{not } f(X, Y) \leftarrow$

→ allow for the “free” introduction of unary and binary literals, provided other rules do not impose extra constraints.

# Unary Rules

E.g.,

$$a(X) \leftarrow f(X, Y_1), \text{ not } g(X, Y_2), h(X, Y_2), Y_1 \neq Y_2$$

→ *branching* or *tree* structure.

→ positive connection between each *node*  $X$  and a successor  $Y_i$ .

## Binary Rules

E.g.,

$$f(X, Y) \leftarrow a(X), \text{ not } b(X), g(X, Y), c(Y)$$

# Constraints

$$\leftarrow a(X)$$

or

$$\leftarrow f(X, Y)$$

## From OASP to 2ATA: Example

Program  $P$  with rule  $r : a(X) \leftarrow a(X)$ . Construct 2ATA  $A_{a,P} = (\Sigma, Q, \delta, q_0, \Omega)$ :

$\Sigma$

$$2^{\{a\}} = \{\emptyset, \{a\}\}$$

$\delta$

$$\begin{aligned}\delta(q_0, \emptyset) &= a \in \emptyset \wedge (0, q_1) \\ &= \mathbf{false}\end{aligned}$$

$$\begin{aligned}\delta(q_0, \{a\}) &= a \in \{a\} \wedge (0, q_1) \\ &= (0, q_1)\end{aligned}$$

## From OASP to 2ATA: Example (2)

$$\delta(q_1, \emptyset) = (0, \overline{q_a}) \wedge \bigwedge_{1 \leq i \leq k} (i, q_1)$$

$$= (0, \overline{q_a})$$

$$\delta(q_1, \{a\}) = (0, q_a)$$

$$\delta(q_a, \emptyset) = \mathbf{false}$$

$$\delta(q_a, \{a\}) = (0, q_r)$$

$$\delta(q_r, \emptyset) = \delta(q_r, \{a\}) = (0, q_a)$$

$$\delta(\overline{q_a}, \emptyset) = (0, \overline{q_r})$$

$$\delta(\overline{q_a}, \{a\}) = \mathbf{false}$$

$$\delta(\overline{q_r}, \emptyset) = \delta(\overline{q_r}, \{a\}) = (0, \overline{q_a})$$

## From OASP to 2ATA: Example (3)

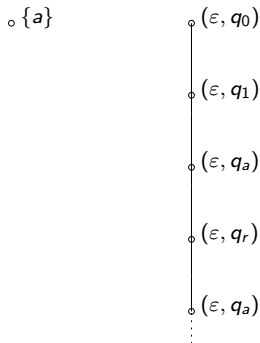
### Acceptance Condition

$$(\{q_a\}, \{q_0, q_1, q_a, q_r, \overline{q_a}, \overline{q_r}\})$$

*Positive states ( $q_a$ ) cannot appear infinitely: not minimal.*

## From OASP to 2ATA: Example (4)

$a$  is not satisfiable w.r.t.  $a(X) \leftarrow a(X)$ : there is no infinite tree that is accepted by  $A_{a,P}$ .



→  $q_a$  is visited infinitely often.

# Decidability of Conceptual Logic Programs

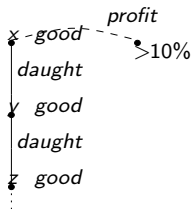
Satisfiability checking w.r.t. Conceptual Logic Programs is decidable and in EXPTIME.

# Forest Logic Programs

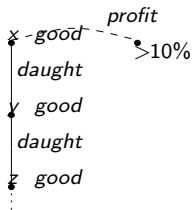
- ▶ Allow for constants: *forest logic programs*.
- ▶ Loss of tree-model property: *forest-model* property.
- ▶ Decidability directly via 2ATA no longer possible.
- ▶ Alternative approach: identify fragment of forest forest logic programs for which satisfiability checking can be reduced to finite answer set programming – *local forest logic programs*.

## Forest Logic Programs: Example

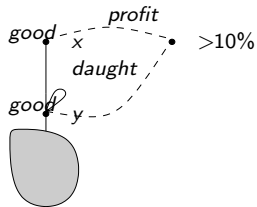
$profit(C, P) \vee not\ profit(C, P) \leftarrow$   
 $daught(C, D) \vee not\ daught(C, D) \leftarrow$   
 $good(C) \leftarrow profit(C, >10\%)$   
 $good(C) \leftarrow daught(C, D), not\ bad(D)$   
 $bad(C) \leftarrow not\ good(C)$



# Forest Logic Programs: Cutting

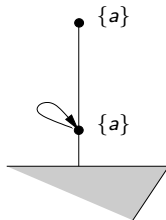
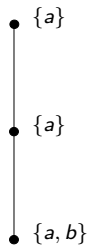


cut version:



# Cutting Problem

$$\begin{aligned} a(X) &\leftarrow f(X, Y), a(Y) \\ a(X) &\leftarrow b(X) \\ b(X) \vee \text{not } b(X) &\leftarrow \\ f(X, Y) \vee \text{not } f(X, Y) &\leftarrow \end{aligned}$$



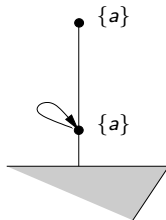
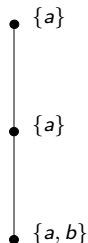
# Cutting Problem

$a(X) \leftarrow f(X, Y), a(Y)$

$a(X) \leftarrow b(X)$

$b(X) \vee \text{not } b(X) \leftarrow$

$f(X, Y) \vee \text{not } f(X, Y) \leftarrow$



# Local Forest Logic Programs

$$a(X) \leftarrow f(X, Y), \text{ not } a(Y)$$

instead of

$$a(X) \leftarrow f(X, Y), a(Y)$$

- ▶ reduction to finite answer set programming possible for local forest logic programs, but in  $2\text{-EXPTIME}^{\text{NEXPTIME}}$ .
- ▶ variants: semi-local, acyclic, with arbitrary rule part (grounded with constants from the program only).

# Guarded Programs

- ▶ From OASP to Fixed Point Logic
- ▶ Decidability: From Guarded OASP to Guarded Fixed Point Logic

# Guarded OASP

A rule  $r : \alpha \leftarrow \beta$  is *guarded* if there is an atom  $\gamma_b \in \beta^+$  such that  $\text{vars}(r) \subseteq \text{vars}(\gamma_b)$ .

A program  $P$  is a *guarded program* if every non-free rule in  $P$  is guarded.

# Guarded OASP via $\mu$ GF

## Theorem

*Satisfiability checking w.r.t. guarded programs is in 2-EXPTIME.*

# Simulating DL

- ▶ DL knowledge base

$$\exists man\_in.Co \sqcap \exists has\_pr \sqsubseteq Product$$

*If something is manufactured in some country and it has a price then it is a product.*

- ▶ Corresponding Conceptual Logic Program:

$$\leftarrow (\exists man\_in.Co \sqcap \exists has\_pr)(X), not Product(X)$$

## Simulating DL (2)

- ▶ Definition of predicates corresponding to concept expressions:

$$\begin{aligned}(\exists \textit{man\_in.Co} \sqcap \exists \textit{has\_pr})(X) &\leftarrow (\exists \textit{man\_in.Co})(X), (\exists \textit{has\_pr})(X) \\(\exists \textit{man\_in.Co})(X) &\leftarrow \textit{man\_in}(X, Y), \textit{Co}(Y) \\(\exists \textit{has\_pr})(X) &\leftarrow \textit{has\_pr}(X, Y)\end{aligned}$$

- ▶ Free predicates correspond to concept names:

$$\begin{aligned}\textit{Product}(X) \vee \textit{not Product}(X) &\leftarrow \\ \textit{Co}(X) \vee \textit{not Co}(X) &\leftarrow \\ \textit{man\_in}(X, Y) \vee \textit{not man\_in}(X, Y) &\leftarrow \\ \textit{has\_pr}(X, Y) \vee \textit{not has\_pr}(X, Y) &\leftarrow\end{aligned}$$

# Conclusions

- ▶ From Closed to Open
- ▶ From Undecidable to Decidable:
  - ▶ Conceptual Logic Programs (via 2ATA)
  - ▶ Local Forest Logic Programs (via closed ASP)
  - ▶ Guarded Programs (via FPL)
- ▶ Applications: DL, DL with rules, CTL, Datalog LITE
- ▶ Advantages of OASP over DLs: nonmonotonism, natural rule-based presentation