

Reasoning with Forest Logic Programs Using Fully Enriched Automata

Cristina Feier¹ Thomas Eiter²

¹ Department of Computer Science, University of Oxford, Oxford UK

² Institute of Information Systems, Vienna University of Technology, Vienna Austria

Ontologies and Logic Programming for Query Answering
Buenos Aires, Argentina

Introduction

Forest Logic Programs:

- decidable fragment of Open Answer Set Programming
- non-monotonicity and rule-based syntax
- open domain semantics
- can simulate reasoning with the expressive DL *SHOQ*

Previous work:

- non-deterministic tableau algorithms: 2NEXPTIME, NEXPTIME running time
- exact complexity characterization still open

Current work:

- encoding of reasoning with FoLPs into emptiness checking of fully enriched automata \implies EXPTIME \implies worst-case optimal

Answer Set Programming

non-monotonic language: combinatoric problems, reasoning about actions, planning

Syntax

- rules of the form: $\alpha \leftarrow \beta$, with α a disjunction of regular literals and β a conjunction of literals
- literals: *not* **a**, **a**, $s \neq t$

Semantics: the Stable Model Semantics

- implicit domain: Herbrand universe
- *not* operator interpreted as *negation as failure*
- I (set of ground atoms) is a model of (ground) P iff it is a *minimal model* of: $P^I = \{\alpha^+ \leftarrow \beta^+ \mid \alpha \leftarrow \beta \in P, I \models \alpha^-, I \not\models \beta^-\}$
(the Gelfond-Lifschitz Reduct)

Closed vs. Open World Reasoning in ASP

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

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

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

→ answer set: $\{pass(john)\}$.

→ *fail* is not satisfiable:

- assume the presence of anonymous objects – *open domains*
- e.g. with universe $\{john, x\}$, *fail* becomes satisfiable

Open Answer Set Programming (OASP)

Enhancing Answer Set Programming with open domains:

Syntax

same as the syntax of *function-free* Answer Set Programming

Semantics (OASP)

- (U, M) is an *open answer set* of an OASP (FoLP) P , iff $U \supseteq \text{cts}(P)$ and M is an answer set of P_U

When $U = \{\text{john}, x\}$, P_U :

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

$M = \{\text{pass}(\text{john}), \text{fail}(x)\}$ is an answer set of P_U : \rightsquigarrow
 $(\{\text{john}, x\}, \{\text{pass}(\text{john}), \text{fail}(x)\})$ is an open answer set!

Forest Logic Programs

OASP is undecidable: syntactical restrictions to achieve decidability;

Forest Logic Programs

- allow only for unary and binary predicates

Forest Logic Programs

OASP is undecidable: syntactical restrictions to achieve decidability;

Forest Logic Programs

- allow only for unary and binary predicates
- tree-shaped rules:

$$\begin{aligned} r_1 : \text{LitLover}(X) &\leftarrow \text{read}(X, Y_1), \text{read}(X, Y_2), \\ &\quad \text{Novel}(Y_1), \text{Novel}(Y_2), Y_1 \neq Y_2 \\ r_2 : \text{Novel}(X) &\leftarrow \text{wrBy}(X, Y), \text{Novelist}(Y) \\ r_3 : \text{Novelist}(X) &\leftarrow \text{wrote}(X, Y), \text{Novel}(Y) \end{aligned}$$

Forest Logic Programs

OASP is undecidable: syntactical restrictions to achieve decidability;

Forest Logic Programs

- allow only for unary and binary predicates
- tree-shaped rules: *forest model property*
- a special type of unsafe rules: *free rules*

$$\begin{aligned} r_1 : \quad & \text{LitLover}(X) \leftarrow \text{read}(X, Y_1), \text{read}(X, Y_2), \\ & \quad \quad \quad \text{Novel}(Y_1), \text{Novel}(Y_2), Y_1 \neq Y_2 \\ r_2 : \quad & \text{Novel}(X) \leftarrow \text{wrBy}(X, Y), \text{Novelist}(Y) \\ r_3 : \quad & \text{Novelist}(X) \leftarrow \text{wrote}(X, Y), \text{Novel}(Y) \\ r_4 : \quad & \text{read}(X, Y) \vee \text{not read}(X, Y) \leftarrow \\ r_5 : \quad & \text{wrBy}(X, Y) \vee \text{not wrBy}(X, Y) \leftarrow \\ r_6 : \quad & \text{wrote}(X, Y) \vee \text{not wrote}(X, Y) \leftarrow \end{aligned}$$

Forest Logic Programs

OASP is undecidable: syntactical restrictions to achieve decidability;

Forest Logic Programs

- allow only for unary and binary predicates
- tree-shaped rules: *forest model property*
- a special type of unsafe rules: *free rules*
- **facts**

$$\begin{aligned} r_1 : \quad & \text{LitLover}(X) \leftarrow \text{read}(X, Y_1), \text{read}(X, Y_2), \\ & \quad \quad \quad \text{Novel}(Y_1), \text{Novel}(Y_2), Y_1 \neq Y_2 \\ r_2 : \quad & \text{Novel}(X) \leftarrow \text{wrBy}(X, Y), \text{Novelist}(Y) \\ r_3 : \quad & \text{Novelist}(X) \leftarrow \text{wrote}(X, Y), \text{Novel}(Y) \\ r_4 : \quad & \text{read}(X, Y) \vee \text{not read}(X, Y) \leftarrow \\ r_5 : \quad & \text{wrBy}(X, Y) \vee \text{not wrBy}(X, Y) \leftarrow \\ r_6 : \quad & \text{wrote}(X, Y) \vee \text{not wrote}(X, Y) \leftarrow \\ f_1 : \quad & \text{Novel}(a) \leftarrow \\ f_2 : \quad & \text{Novelist}(b) \leftarrow \end{aligned}$$

Forest model property

A unary predicate is satisfiable iff it is satisfied by a forest-shaped model

$r_1 : \text{LitLover}(X) \leftarrow \text{read}(X, Y_1), \text{read}(X, Y_2),$
 $\text{Novel}(Y_1), \text{Novel}(Y_2), Y_1 \neq Y_2.$

$r_2 : \text{Novel}(X) \leftarrow \text{wrBy}(X, Y), \text{Novelist}(Y).$

$r_3 : \text{Novelist}(X) \leftarrow \text{wrote}(X, Y), \text{Novel}(Y).$

$r_4 : \text{read}(X, Y) \vee \text{not read}(X, Y) \leftarrow .$

$r_5 : \text{wrBy}(X, Y) \vee \text{not wrBy}(X, Y) \leftarrow .$

$r_6 : \text{wrote}(X, Y) \vee \text{not wrote}(X, Y) \leftarrow .$

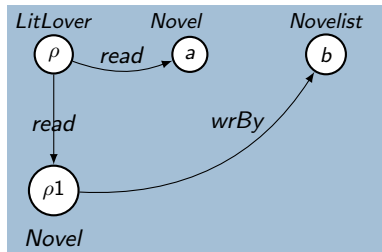
$f_1 : \text{Novel}(a).$

$f_2 : \text{Novelist}(b).$

(U, M) with:

- $U = \{\rho, \rho1, a, b\}$, and
- $M = \{\text{LitLover}(\rho), \text{Novel}(a), \text{read}(\rho, \rho1), \dots\}$

is a forest model.



Forest model property

A unary predicate is satisfiable iff it is satisfied by a forest-shaped model

$r_1 : \mathbf{LitLover}(X) \leftarrow read(X, Y_1), read(X, Y_2),$
 $Novel(Y_1), Novel(Y_2), Y_1 \neq Y_2.$

$r_2 : Novel(X) \leftarrow wrBy(X, Y), Novelist(Y).$

$r_3 : Novelist(X) \leftarrow wrote(X, Y), Novel(Y).$

$r_4 : read(X, Y) \vee not\ read(X, Y) \leftarrow .$

$r_5 : wrBy(X, Y) \vee not\ wrBy(X, Y) \leftarrow .$

$r_6 : wrote(X, Y) \vee not\ wrote(X, Y) \leftarrow .$

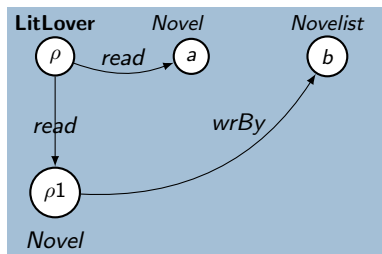
$f_1 : Novel(a).$

$f_2 : Novelist(b).$

(U, M) with:

- $U = \{\rho, \rho1, a, b\}$, and
- $M = \{\mathbf{LitLover}(\rho), Novel(a), read(\rho, \rho1), \dots\}$

is a forest model.



The degree of a FoLP P

The maximum out-degree of a forest model

- for a unary rule r , $degree(r)$: the number k of successor variables which appear in r
- for a unary predicate p ,
 $degree(p) = \max\{degree(r) \mid p \in preds(head(r))\}$
- for a FoLP P , $degree(P) = \sum_{p \in upr(P)} degree(p)$

$r_1 : LitLover(X) \leftarrow read(X, Y_1), read(X, Y_2),$
 $Novel(Y_1), Novel(Y_2), Y_1 \neq Y_2.$

$r_2 : Novel(X) \leftarrow wrBy(X, Y), Novelist(Y).$

$r_3 : Novelist(X) \leftarrow wrote(X, Y), Novel(Y).$

$r_4 : read(X, Y) \vee not\ read(X, Y) \leftarrow .$

$r_5 : wrBy(X, Y) \vee not\ wrBy(X, Y) \leftarrow .$

$r_6 : wrote(X, Y) \vee not\ wrote(X, Y) \leftarrow .$

$f_1 : Novel(a).$

$f_2 : Novelist(b).$

$degree(r_1) = degree(LitLover) = 2$

$degree(r_2) = degree(Novel) = 1$

$degree(r_3) = degree(Novelist) = 1$

$degree(P) = 4$

Alternative Characterization for ASP Semantics [Fages90]

Well-supported Models of a Ground Logic Program P

I is a well-supported model of P iff there is a partial well-founded order \prec on I s.t for every $A \in I$, there exists a rule $R \in P$:

$A \leftarrow B_1, \dots, B_n, \text{not } C_1, \dots, \text{not } C_m$ s.t.:

- $B_i \in I$, for every $1 \leq i \leq n$, and $C_j \notin I$, for every $1 \leq j \leq m$ (*supportedness*).
- $B_i \prec A$, for every $1 \leq i \leq n$ (*well-supportedness*).

Equivalence of Semantics

Well-supported models coincide with the stable models (answer sets) for LPs.

Forest Models are Well-Supported

$r_1 : LitLover(X) \leftarrow read(X, Y_1), read(X, Y_2),$
 $Novel(Y_1), Novel(Y_2), Y_1 \neq Y_2.$

$r_2 : Novel(X) \leftarrow wrBy(X, Y), Novelist(Y).$

$r_3 : Novelist(X) \leftarrow wrote(X, Y), Novel(Y).$

$r_4 : read(X, Y) \vee not\ read(X, Y) \leftarrow .$

$r_5 : wrBy(X, Y) \vee not\ wrBy(X, Y) \leftarrow .$

$r_6 : wrote(X, Y) \vee not\ wrote(X, Y) \leftarrow .$

$f_1 : Novel(a).$

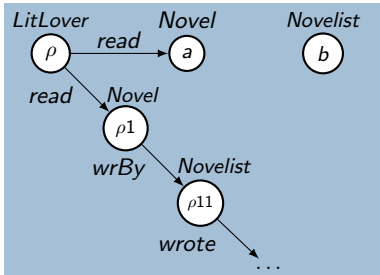
$f_2 : Novelist(b).$

(U, M) with:

- $U = \{\rho, \rho1, \dots, a, b\}$, and
- $M = \{LitLover(\rho), Novel(a), Novelist(b), read(\rho, \rho1), Novel(\rho1), \dots\}$

is not a forest model.

Challenge: Construct well-supported models!



Fully Enriched Automata

$B^+(Y)$ the set of positive Boolean formulas over Y

$D_b = \{\langle 0 \rangle, \langle 1 \rangle, \dots, \langle b \rangle\} \cup \{[0], [1], \dots, [b]\} \cup \{-1, \varepsilon, \langle root \rangle, [root]\}$

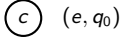
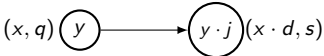
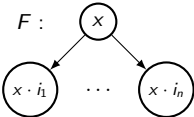
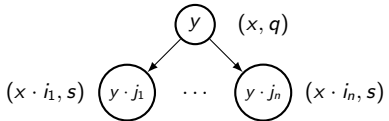
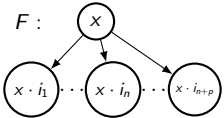
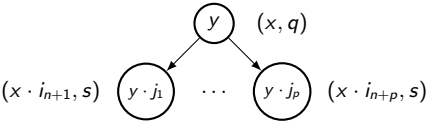
$A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$:

- Σ is a finite input alphabet
- $b > 0$ is a counting bound
- Q is a finite set of states
- $\delta : Q \times \Sigma \rightarrow B^+(D_b \times Q)$ - the transition function
- $q_0 \in Q$ - the initial state
- $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_k\}$, where $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \dots \subseteq \mathcal{F}_k = Q$ is a *parity acceptance condition*
 - ▶ k : the *index* of the automaton.

Run of a Fully Enriched Automata

$A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$: a FEA; (F, V) : a labeled forest with N_F the set of nodes and $V : N_F \rightarrow \Sigma$


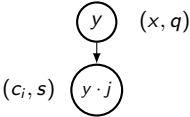
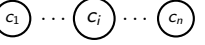
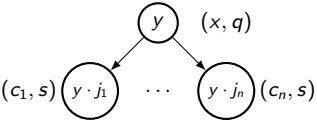
A run of A on (F, V) : a labeled tree (T_c, r) with $r : T_c \rightarrow N_F \times Q$:

Condition	Run (T_c, r)	
e - some root in F		
$d = -1, \varepsilon$ $x \cdot -1 \rightarrow x$ $x = x \cdot \varepsilon$		
$\delta(q, V(x)) = \Theta$ $\exists S. S \subseteq D_B \times Q$ $\wedge S$ satisfies Θ $\forall (d, s) \in S :$	$d = \langle n-1 \rangle$ $F :$ 	
$d = [n]$ $F :$ 		

Run of a Fully Enriched Automata

$A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$: a FEA; (F, V) : a labeled forest with N_F the set of nodes and $V : N_F \rightarrow \Sigma$

A run of A on F, V : a labeled tree (T_c, r) with $r : T_c \rightarrow N_F \times Q$:

Condition	Run (T_c, r)
$\delta(q, V(x)) = \Theta$ $\exists S. S \subseteq D_B \times Q$ $\wedge S$ satisfies Θ $\forall (d, s) \in S :$	$d = \langle \text{root} \rangle$ $F :$  
$d = [\text{root}]$ $F :$ 	

Acceptance Condition

$A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ a FEA with $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \dots, \mathcal{F}_k\}$

(T_c, r) a run of A on (F, V)

- an infinite path in (T_c, r) is accepting if the minimum i s.t. a state q in \mathcal{F}_i occurs infinitely often on the path is even
- (T_c, r) is accepting if each infinite path in T_c is accepting
- A accepts (F, V) iff there exists an accepting run of A on (F, V)

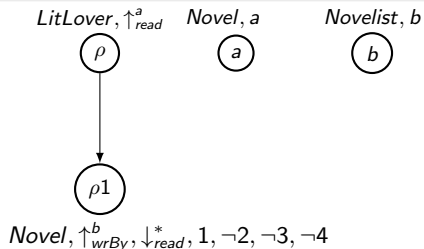
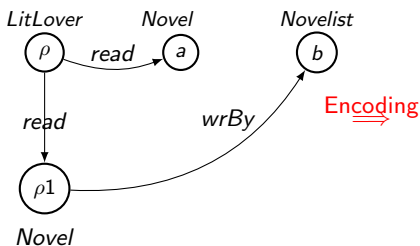
The language of A , denoted $\mathcal{L}(A)$, is the set of forests accepted by A .

Given a FEA $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F} \rangle$ with n states and index k , deciding whether $\mathcal{L}(A) = \emptyset$ is possible in time $(b+2)^{\mathcal{O}(n^3 \cdot k^2 \cdot \log k \cdot \log b^2)}$.

A Class of FEAs $A_{\rho, \theta}^{P, P}$ which accept Forest Models

P : a FoLP, p : a unary predicate, $d = \text{degree}(P)$

- $\rho \in \text{cts}(P) \cup \{x\}$
- $\theta : \text{cts}(P) \cup \{\rho\} \rightarrow 2^{\text{upr}(P) \cup \text{cts}(P) \cup \{\rho\}}$:
 - ▶ $o_i \in \theta(o_i)$, and $o_j \notin \theta(o_i)$, for every $o_i, o_j \in \text{cts}(P) \cup \{\rho\}$, s. t. $o_i \neq o_j$.
 - ▶ $p \in \theta(\rho)$.
- $\text{PAT}_P = \{*\} \cup \text{cts}(P)$: $t \mapsto pt$ iff $t = pt$, when t is a constant;
- $\Sigma = 2^S$, where $S = \text{upr}(P) \cup \text{cts}(P) \cup \{\rho\} \cup \{\uparrow_f^o \mid f \in \text{bpr}(P)\} \cup \{\downarrow_f^t \mid f \in \text{bpr}(P), t \in \text{PAT}_P\} \cup \{1, \dots, d\}$.



Definition of $A_{\rho, \theta}^{P, P}$ (ctd.)

$Q = Q_i \cup Q_+ \cup Q_-$, with:

- $Q_i = \{q_0, q_1\} \cup \{q_o \mid o \in \text{cts}(P) \cup \{\rho\}\} \cup \{q_{-k} \mid 1 \leq k \leq d\}$,
- $Q_+ = \{q_{t,a}, q_{t,r_a}, q_{t_1,t_2,u}, q_{t_1,t_2,r_f}, q_{k,t,*,u}\}$, where $t, t_1, t_2 \in \text{PAT}_P$, $a \in \text{upr}(P)$, $f \in \text{bpr}(P)$, u is of the form a, f , *not a* or *not f*, $1 \leq k \leq d$, $r_a \in \text{urul}(P)$, $r_f \in \text{brul}(P)$,
 - ▶ motivate the presence of atoms in an open answer set
- $Q_- = \{q_{\overline{t,a}}, q_{\overline{t,r_a}}, q_{\overline{t_1,t_2,u}}, q_{\overline{t_1,t_2,r_f}}, q_{\overline{k,t,*,u}}\}$.
 - ▶ motivate the absence of atoms in an open answer set

Initial transition:

- $\delta(q_0, \sigma) = \bigwedge_{o \in \text{cts}(P) \cup \{\rho\}} (\langle \text{root} \rangle, q_o)$

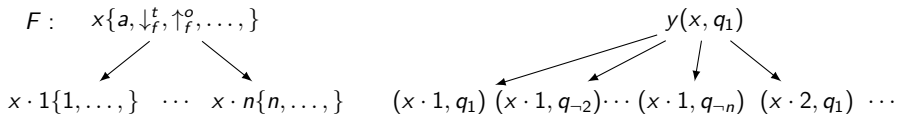
Root transitions:

- $\delta(q_o, \sigma) = o \in \sigma \wedge \bigwedge_{o' \in \text{cts}(P) \cup \{\rho\} - \{o\}} o' \notin \sigma \wedge \bigwedge_{a \in \theta(o)} (\varepsilon, q_{o,a}) \wedge \bigwedge_{a \notin \theta(o)} (\varepsilon, q_{\overline{o,a}}) \wedge \bigwedge_{\uparrow_f^{o'} \in \theta(o)} (\varepsilon, q_{o,o',f}) \wedge \bigwedge_{\uparrow_f^{o'} \notin \theta(o)} (\varepsilon, q_{\overline{o,o',f}}) \wedge ([0], q_1)$

Definition of $A_{\rho, \theta}^{P, P}$ (ctd.)

Traversing the forest:

- $\delta(q_1, \sigma) = ([0], q_1) \wedge \bigwedge_{1 \leq k \leq d} ([1], q_{-k}) \wedge \bigwedge_{a \in \sigma} (\varepsilon, q_{*,a}) \wedge \bigwedge_{a \notin \sigma} (\varepsilon, q_{*,\bar{a}}) \wedge \bigwedge_{\downarrow_f^t \in \sigma} (\varepsilon, q_{t,*,f}) \wedge \bigwedge_{\downarrow_f^t \notin \sigma} (\varepsilon, q_{t,*,\bar{f}}) \wedge \bigwedge_{\uparrow_f^o \in \sigma} (\varepsilon, q_{*,o,f}) \wedge \bigwedge_{\uparrow_f^o \notin \sigma} (\varepsilon, q_{*,o,\bar{f}})$
- $\delta(q_{-k}, \sigma) = k \notin \sigma$



Justifying the presence of unary/binary predicates in the label:

- $\delta(q_{*,a}, \sigma) = a \in \sigma \wedge \left(\text{free}(a) \vee \bigvee_{r_a: a(s) \leftarrow \beta \in P} (\varepsilon, q_{*,r_a}) \right)$
- $\delta(q_{o,a}, \sigma) = o \notin \sigma \vee a \in \theta(o) \wedge \left(\text{free}(a) \vee \bigvee_{r_a: a(s) \leftarrow \beta \in P, s \mapsto o} (\varepsilon, q_{o,r_a}) \right)$
- $\delta(q_{t,*,f}, \sigma) = \downarrow_f^t \in \sigma \wedge \left(\text{free}(f) \vee \bigvee_{r_f: f(s,v) \leftarrow \beta \in P, s \mapsto t, v \mapsto *} (\varepsilon, q_{t,*,r_f}) \right)$
- $\delta(q_{t,o,f}, \sigma) = \uparrow_f^o \in \sigma \wedge \left(\text{free}(f) \vee \bigvee_{r_f: f(s,v) \leftarrow \beta \in P, s \mapsto t, v \mapsto o} (\varepsilon, q_{t,o,r_f}) \right)$

Definition of $A_{\rho,\theta}^{P,P}$ (ctd.)

$r_a : a(s) \leftarrow \beta(s), (\gamma_i(s, v_i), \delta_i(v_i))_{1 \leq i \leq m}, \psi$ - unary rule in P .

$J_{r_a} : a$ multiset $\{j_i \mid 1 \leq i \leq m, j_i \in \{1, \dots, d\} \cup \text{cts}(P)\}$:

- for every $j_i \in J_{r_a}$, $v_i \in \text{cts}(P)$ implies $j_i = v_i$, and
- for every $j_i, j_l \in J_{r_a}$, $v_i \neq v_l \in \psi$ implies $j_i \neq j_l$.

\mathcal{MJ} : the set of all such multisets

$r_a : a(X) \leftarrow f(X, Y_1), b(Y_1), g(X, Y_2), b(Y_2), h(X, Y_3), b(Y_3), Y_1 \neq Y_2,$
 $\text{degree}(P) = 3, \text{cts}(P) = \{c\}$

$J_{r_a} : \{1, c, 1\}, \{1, 2, 3\}, \text{etc.}$

Check that the body of r_a is satisfiable:

- $\delta(q_{t,r_a}, \sigma) =$
 $\bigwedge_{u \in \beta} (\varepsilon, q_{*,t,u}) \wedge \bigvee_{J_{r_a} \in \mathcal{MJ}} \left(\bigwedge_{k=1}^d \bigwedge_{j_i=k, j_i \in J_{r_a}} \bigwedge_{u \in \gamma_i \cup \delta_i} (\langle 0 \rangle, q_{k,t,*,u}) \wedge \right.$
 $\left. \bigwedge_{o \in \text{cts}(P)} \bigwedge_{j_i=o, j_i \in J_{r_a}} \bigwedge_{u \in \gamma_i \cup \delta_i} (\varepsilon, q_{t,o,u}) \right)$

Check that u is (is not) in the label of the k -th successor of a given node:

- $\delta(q_{k,t_1,t_2,u}, \sigma) = k \in \sigma \wedge \bigwedge_{j \neq k} j \notin \sigma \wedge (\varepsilon, q_{t_1,t_2,u})$

Definition of $A_{\rho,\theta}^{P,P}$ (ctd.)

$$\delta(q_{t_1,t_2,u}, \sigma) = \begin{cases} (\varepsilon, q_{*,a}), & \text{if } t_2 = * \text{ and } u = a & (1) \\ ([root], q_{o,a}), & \text{if } t_2 = o \text{ and } u = a & (2) \\ a \notin \sigma, & \text{if } t_2 = * \text{ and } u = \textit{not } a & (3) \\ a \notin \theta(o), & \text{if } t_2 = o \text{ and } u = \textit{not } a & (4) \\ \downarrow_f^t \notin \sigma, & \text{if } t_2 = * \text{ and } u = \textit{not } f & (5) \\ \uparrow_f^o \notin \theta(o), & \text{if } t_2 = o \text{ and } u = \textit{not } f & (6) \end{cases}$$

$r_f : f(s, v) \leftarrow \beta(s), \gamma(s, v), \delta(v)$ - binary rule in P

- $\delta(q_{t,*,r_f}, \sigma) = \bigwedge_{u \in \beta} (-1, q_{*,t,u}) \wedge \bigwedge_{u \in \gamma \cup \delta} (\varepsilon, q_{t,*,u})$
- $\delta(q_{t,o,r_f}, \sigma) = \bigwedge_{u \in \beta} (\varepsilon, q_{*,t,u}) \wedge \bigwedge_{u \in \gamma \cup \delta} (\varepsilon, q_{t,o,u})$

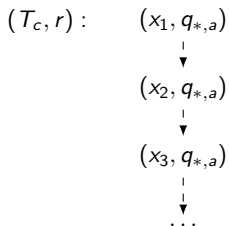
Similar transition rules for negative states.

Definition of $A_{\rho, \theta}^{P, P}$ (ctd.)

Parity Acceptance Condition:

- $\mathcal{F}_1 = \{q_{t,a}, q_{t_1,t_2,f} \mid a \in \text{upr}(P), f \in \text{bpr}(P), t, t_1, t_2 \in \text{PAT}_P\}$
- $\mathcal{F}_2 = Q$

Infinite paths should not revisit a state of the type $q_{t,a}$ or $q_{t_1,t_2,f}$ infinitely often: well-supportedness!



$a(x_1)$ depends on $a(x_2)$; $a(x_2)$ depends on $a(x_3)$; and so on...

Properties of the Encoding

Let P be a FoLP and p be a unary predicate symbol. Then, p is satisfiable with respect to P iff there exists an automaton $A_{\rho,\theta}^{p,P}$ such that $\mathcal{L}(A_{\rho,\theta}^{p,P}) \neq \emptyset$.

Satisfiability checking of unary predicates with respect to FoLPs is EXPTIME-complete.

f-hybrid KBs: pairs (Σ, P)

- Σ a SHOQ kb, P a FoLP: no restriction on signature sharing
- a unary predicate p is satisfiable w.r.t. (Σ, P) iff it is satisfiable w.r.t. $\Theta(\Sigma) \cup P$, where Θ is a polynomial and modular translation from SHOQ to FoLPs.

Satisfiability checking of unary predicates with respect to f-hybrid KBs is EXPTIME-complete.

Conclusions

The result closes an open problem: exact complexity characterization of FoLPs

FEAs - elegant device for encoding

- accept forests as input
- parity acceptance condition to check well-supportedness
- additional addressing and term matching mechanisms needed

Existing work on AND/OR tableau reasoners for CoLPs (FoLPs minus constants):

- how can it be lifted to FoLPs?

Questions?