Reasoning with Forest Logic Programs Using Fully Enriched Automata

Cristina Feier[1] Thomas Eiter[2]

[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 $\mathcal{SHOQ}$

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 $\Longrightarrow$ ExpTime $\Longrightarrow$ 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

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

$\rightarrow$ *ground* the program with all constants (john):

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

$\rightarrow$ answer set: $\{\mathit{pass}(\mathit{john})\}$.
$\rightarrow$ *fail* is not satisfiable:

- assume the presence of anonymous objects – *open domains*
- e.g. with universe $\{\mathit{john}, x\}$, *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 $U \supseteq cts(P)$ and M is an answer set of P_U

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

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

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

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

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*

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

Forest Logic Programs

OASP is undecidable: syntactical restrictions to achieve decidability;

Forest Logic Programs

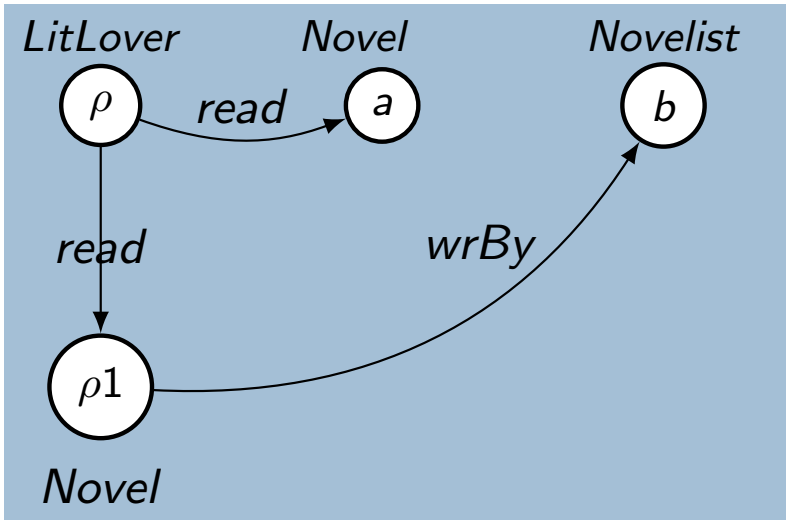
- allow only for unary and binary predicates
- tree-shaped rules: *forest model property*
- a special type of unsafe rules: *free rules*
- facts

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

Forest model property

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

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



(U, M) with:

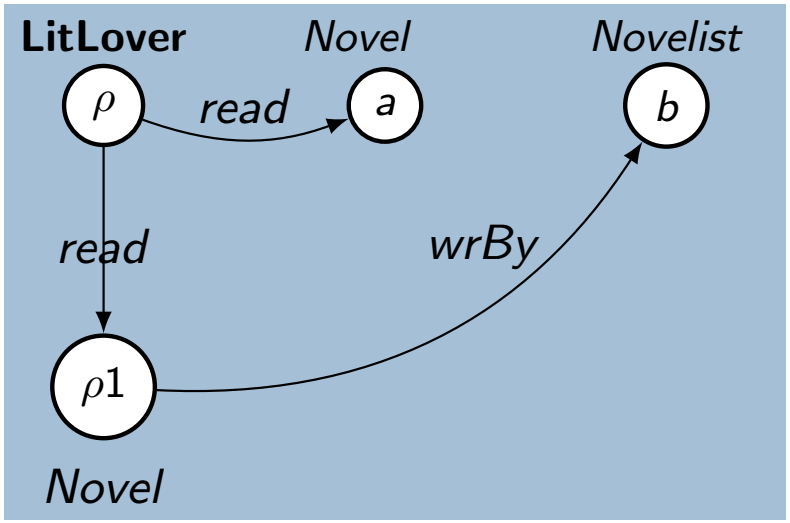
- $U = \{\rho, \rho 1, a, b\}$, and
- $M = \{\mathit{LitLover}(\rho),\ \mathit{Novel}(a),\ \mathit{read}(\rho, \rho 1),\ \ldots\}$

is a forest model.

Forest model property

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

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:

- $U = \{\rho, \rho 1, a, b\}$, and
- $M = \{$**LitLover**(ρ), *Novel*(a), *read*$(\rho, \rho 1)$, $\ldots\}$

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 $R \in P$:
$A \leftarrow B_1, \ldots B_n, not\ C_1, \ldots, not\ C_m$ s.t.:

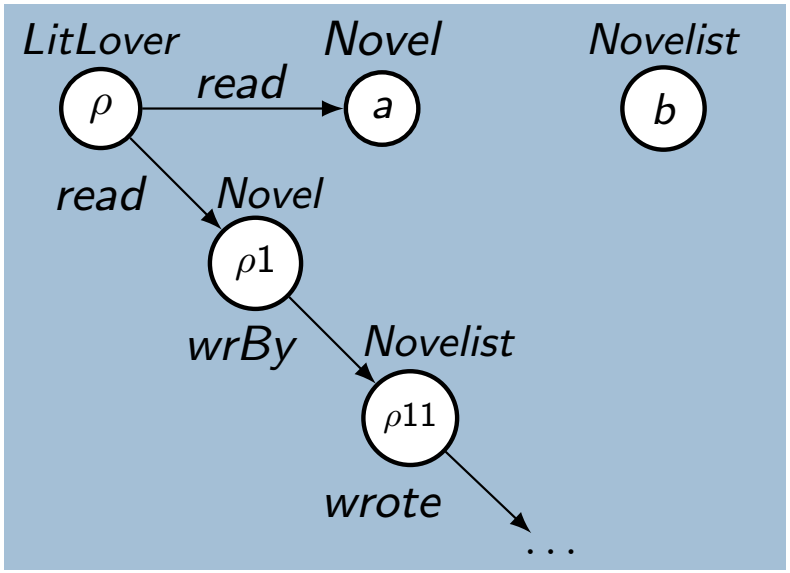
- $B_i \in I$, for every $1 \leqslant i \leqslant n$, and $C_j \notin I$, for every $1 \leqslant j \leqslant m$ (*supportedness*).
- $B_i \prec A$, for every $1 \leqslant i \leqslant n$ (*well-supportedness*).

Equivalence of Semantics

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

Forest Models are Well-Supported

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



(U, M) with:

- $U = \{\rho, \rho 1, \ldots, a, b\}$, and
- $M = \{\mathit{LitLover}(\rho),\ \mathit{Novel}(a), \mathit{Novelist}(b)\ \mathit{read}(\rho, \rho 1), \mathit{Novel}(\rho 1),\ \ldots\}$

is not a forest model.

Challenge: Construct well-supported models!

Fully Enriched Automata

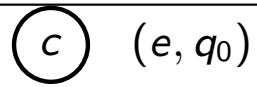
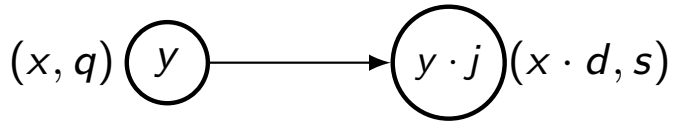
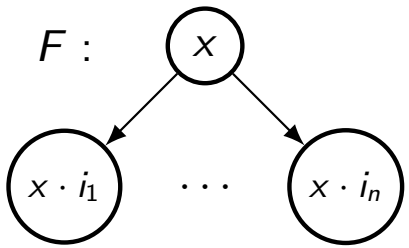
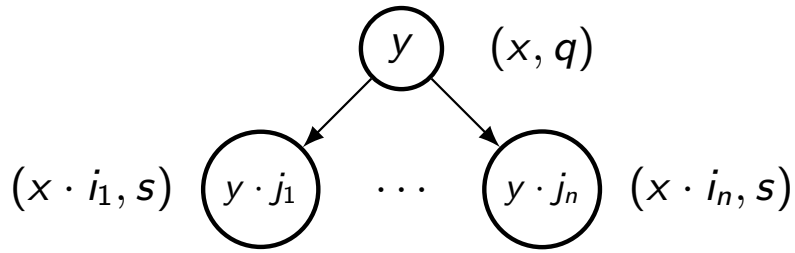
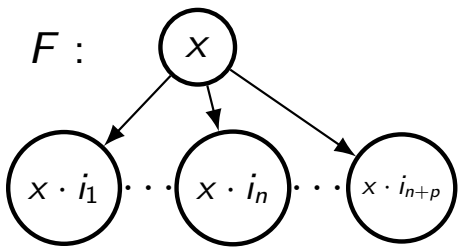
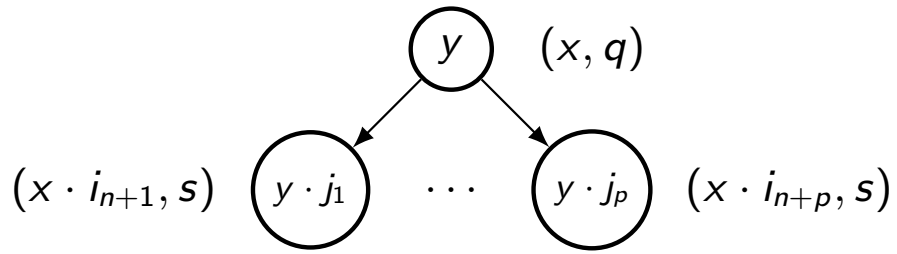
$B^+(Y)$ the set of positive Boolean formulas over Y
$D_b = \{\langle 0\rangle, \langle 1\rangle, \ldots, \langle b\rangle\} \cup \{[0], [1], \ldots, [b]\} \cup \{-1, \varepsilon, \langle root\rangle, [root]\}$
$A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$:

- Σ is a finite input alphabet
- $b > 0$ is a counting bound
- Q is a finite set of states
- $\delta : Q \times \Sigma \to B^+(D_b \times Q)$ - the transition function
- $q_0 \in Q$ - the initial state
- $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$, where $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots \subseteq \mathcal{F}_k = Q$ 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 $V : N_F \to \Sigma$

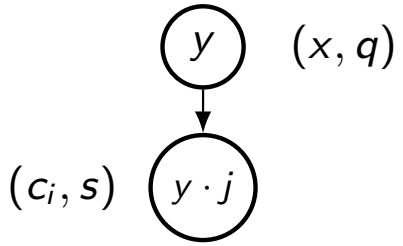
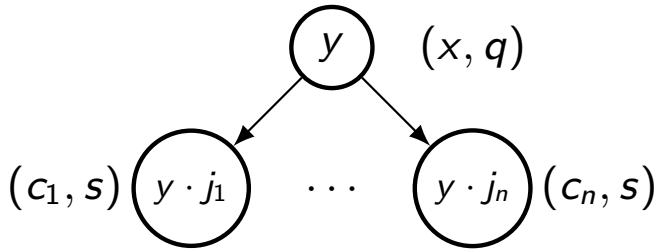
A *run* of A on (F, V): a labeled tree (T_c, r) with $r : T_c \to N_F \times Q$:

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

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

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

Acceptance Condition

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

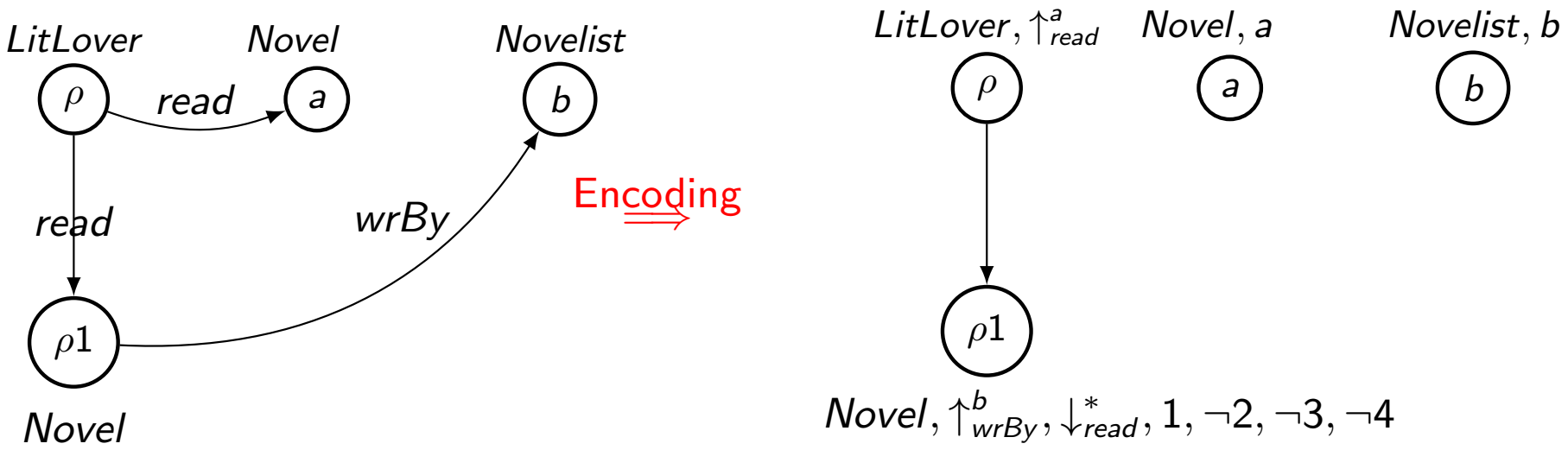
The language of A, denoted $\mathcal{L}(A)$, is the set of forests accepted by A.

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 $A^{p,P}_{\rho,\theta}$ which accept Forest Models

P: a FoLP, p: a unary predicate, $d = degree(P)$

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



Definition of $A^{p,P}_{\rho,\theta}$ (ctd.)

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

- $Q_i = \{q_0, q_1\} \cup \{q_o \mid o \in cts(P) \cup \{\rho\}\} \cup \{q_{\neg k} \mid 1 \leqslant k \leqslant d\}$,
- $Q_+ = \{q_{t,a}, q_{t,r_a}, q_{t_1,t_2,u}, q_{t_1,t_2,r_f}, q_{k,t,*,u}\}$, where $t, t_1, t_2 \in \mathrm{PAT}_P$, $a \in upr(P)$, $f \in bpr(P)$, u is of the form $a, f, not\ a$ or $not\ f$, $1 \leqslant k \leqslant d$, $r_a \in urul(P)$, $r_f \in brul(P)$,
  - 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:

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

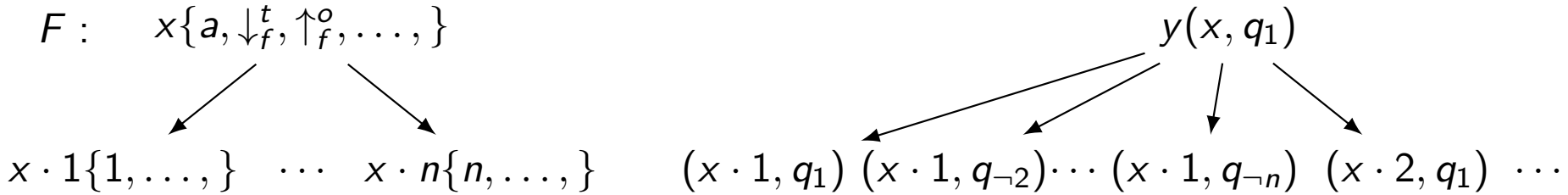
Root transitions:

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

Definition of $A^{p,P}_{\rho,\theta}$ (ctd.)

Traversing the forest:

- $\delta(q_1,\sigma)$=$([0],q_1) \wedge \bigwedge_{1\leqslant k\leqslant d}([1],q_{\neg k}) \wedge \bigwedge_{a\in\sigma}(\varepsilon,q_{*,a}) \wedge \bigwedge_{a\notin\sigma}(\varepsilon,q_{\overline{*,a}}) \wedge \bigwedge_{\downarrow^t_f\in\sigma}(\varepsilon,q_{t,*,f}) \wedge \bigwedge_{\downarrow^t_f\notin\sigma}(\varepsilon,q_{\overline{t,*,f}}) \wedge \bigwedge_{\uparrow^o_f\in\sigma}(\varepsilon,q_{*,o,f}) \wedge \bigwedge_{\uparrow^o_f\notin\sigma}(\varepsilon,q_{\overline{*,o,f}})$
- $\delta(q_{\neg k},\sigma) = k \notin \sigma$



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

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

Definition of $A^{p,P}_{\rho,\theta}$ (ctd.)

$r_a : a(s) \leftarrow \beta(s), (\gamma_i(s, v_i), \delta_i(v_i))_{1 \leqslant i \leqslant m}, \psi$ - unary rule in P.
J_{r_a}: a multiset $\{j_i \mid 1 \leqslant i \leqslant m, j_i \in \{1, \ldots, d\} \cup cts(P)\}$:

- for every $j_i \in J_{r_a}$, $v_i \in cts(P)$ 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

$r_a : a(X) \leftarrow f(X, Y_1), b(Y_1), g(X, Y_2), b(Y_2), h(X, Y_3), b(Y_3), Y_1 \neq Y_2$,
$degree(P) = 3$, $cts(P) = \{c\}$
$J_{r_a} : \{1, c, 1\}, \{1, 2, 3\}$, etc.

Check that the body of r_a is satisfiable:

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

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 $A^{p,P}_{\rho,\theta}$ (ctd.)

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

$r_f : f(s,v) \leftarrow \beta(s), \gamma(s,v), \delta(v)$ - 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.

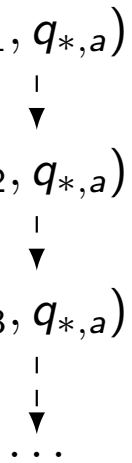
Definition of $A^{p,P}_{\rho,\theta}$ (ctd.)

Parity Acceptance Condition:

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

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

(T_c, r) :



$a(x_1)$ 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 $A^{p,P}_{\rho,\theta}$ such that $\mathcal{L}(A^{p,P}_{\rho,\theta}) \neq \emptyset$.

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

f-hybrid KBs: pairs (Σ, P)

- Σ a $\mathcal{SHOQ}$ 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 $\mathcal{SHOQ}$ 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?