Reasoning with Forest Logic Programs Using Fully Enriched Automata

Cristina Feier
Department of Computer Science
University of Oxford
cristina.feier@cs.ox.ac.uk

Thomas Eiter
Institute of Information Systems
Vienna University of Technology
eiter@kr.tuwien.ac.at

Abstract

Forest Logic Programs (FoLP) are a decidable fragment of Open Answer Set Programming (OASP) which have the forest model property. OASP extends Answer Set Programming (ASP) with open domains—a feature which makes it possible for FoLPs to simulate reasoning with the expressive description logic $\mathcal{SHOQ}$. At the same time, the fragment retains the attractive rule syntax and the non-monotonicity specific to ASP. In the past, several tableaux algorithms have been devised to reason with FoLPs, the most recent of which established a NEXPTIME upper bound for reasoning with the fragment. While known to be EXPTIME-hard, the exact complexity characterization of reasoning with the fragment was still unknown. In this paper we settle this open question by a reduction of reasoning with FoLPs to emptiness checking of fully enriched automata, a form of automata which run on forests, and which are known to be EXPTIME-complete.

Introduction

Open Answer Set Programming (OASP) (Heymans, Van Nieuwenborgh, and Vermeir 2008) extends Answer Set Programming (ASP) (Gelfond and Lifschitz 1988) with an open domain semantics: programs are interpreted w.r.t. arbitrary domains that might contain individuals which do not occur explicitly in the program. This makes it possible to state generic knowledge using OASP. At the same time, OASP inherits from ASP the negation under the stable model semantics. Thus, OASP bridges two important knowledge representation paradigms: the classical First Order Logic (FOL) open world and the non-monotonic rules closed world. It is part of a broad line of research which includes approaches like DL-safe rules (Motik, Sattler, and Studer 2005), $\mathcal{DL}$+*log* (Rosati 2006), dl-programs (Eiter et al. 2008), Description Logic Rules (Krötzsch, Rudolph, and Hitzler 2008), r-hybrid knowledge bases (Rosati 2008), Datalog$^{\pm}$ (Calì, Gottlob, and Lukasiewicz 2009), MKNF^{+} knowledge bases (Motik and Rosati 2010), and Nonmonotonic Existential Rules (Magka, Krötzsch, and Horrocks 2013).

In general, OASP is undecidable. To achieve decidability, several fragments have been defined by imposing syntactical restrictions on the shape of rules. Such a fragment are Forest Logic Programs (FoLP) which enjoy the forest model property: a unary predicate is satisfiable iff it is satisfied by a model that can be represented as a labeled forest, where nodes and arcs are labeled with sets of unary predicates and binary predicates, respectively.

FoLPs are quite an expressive fragment as they allow, for instance, the simulation of standard reasoning tasks (like concept satisfiability and KB consistency) with $\mathcal{SHOQ}$ ontologies (Feier and Heymans 2013). This property of FoLPs led to *f-hybrid KBs*, a combination of rules and ontologies which distinguish themselves among other approaches like dl-safe rules, r-hybrid knowledge bases, or MKNF^{+} knowledge bases, by the fact that they impose no restrictions on the interaction between the signatures of the two components. Such restrictions prevent the need for reasoning with unknown individuals in the rule component. As f-hybrid KBs are based on the simulation of $\mathcal{SHOQ}$ KBs within FoLPs, no such restriction is needed. Conceptual modeling using FoLPs is not restricted to simulating reasoning with $\mathcal{SHOQ}$ KBs: it is also possible to translate *object-role modeling (ORM)* models as sets of FoLP rules, e.g. (Heymans 2006).

As they can simulate reasoning within the Description Logic (DL) $\mathcal{SHOQ}$, it follows that reasoning with FoLPs is EXPTIME-hard. However, the exact complexity characterization of FoLPs was still open. Previously, reasoning with FoLPs was addressed by means of tableau-based algorithms: (Feier and Heymans 2013) described a 2NEXPTIME tableau algorithm, while an improved algorithm which runs in the worst case in NEXPTIME has been described in (Feier 2012). While in the latter work it has been speculated that the non-deterministic tableau algorithm can be determinized in order to lead to an EXPTIME procedure which would be worst-case optimal, the determinization in the case of FoLPs proved elusive. Such a deterministic worst-case optimal algorithm has been devised for CoLPs, which restrict FoLPs to programs without constants, and simple FoLPs, a fragment in which recursion is restricted: the technique does not scale up to FoLPs (see (Feier 2014)).

In this paper, we settle the open question regarding the exact complexity characterization of FoLPs: by using a reduction to emptiness checking of Fully Enriched Automata

(FEAs), we show that satisfiability checking of unary predicates w.r.t. FoLPs is EXPTIME-complete. Hence, reasoning with FoLP rules and $\mathcal{SHOQ}$ ontologies is not harder than reasoning with $\mathcal{SHOQ}$ ontologies themselves.

Fully enriched automata have been introduced in (Bonatti et al. 2008) as a tool to reason with hybrid graded μ-calculus, which extends μ-calculus with graded modalities and nominals. They offer an elegant device for our encoding as they accept forests as inputs and also feature a parity acceptance condition that is useful in distinguishing well-supported models (Fages 1991), a fundamental characteristic of (open) answer sets. However, FoLPs exhibit a specific form of the forest model property, in which every node can point back to any root of the forest, and as such the encoding is not without its challenges.

The automata-based method has been previously applied to reason with CoLPs (Heymans, Van Nieuwenborgh, and Vermeir 2006): satisfiability checking of unary predicates w.r.t. a CoLP has been reduced to non-emptiness checking of two-way alternating tree automata (2ATA) (Vardi 1998). 2ATAs have also been used to check consistency of normal bidirectional ASP programs (bd-programs) (Eiter and Šimkus 2009), which are a decidable fragment of ASP extended with function symbols that also exhibit the tree model property. In the context of DL, 2ATAs have been employed to check concept satisfiability (Calvanese, Giacomo, and Lenzerini 2002) and satisfiability of $\mathcal{ALCQIb}_{reg}$ KBs (Calvanese, Eiter, and Ortiz 2007)–in the latter case, canonical models are forest-shaped, and as such they were encoded as trees in order to be processed using 2ATAs. In the case of FoLPs, it is not clear how such an encoding would work due to the special form of their forest model property. Finally, FEAs were used to encode satisfiability checking of $\mathcal{ZOIQ}$ concepts (Calvanese, Eiter, and Ortiz 2009).

Preliminaries

We start by introducing the open answer set syntax and semantics (Heymans, Van Nieuwenborgh, and Vermeir 2008). We assume countably infinite disjoint sets of constants, variables, and predicate symbols. Terms and atoms are defined as usual. We refer to an atom where the predicate symbol is unary or binary, as a unary or binary atom, resp. A *literal* is an atom a or a negated atom $not\ a$. We allow for *inequality literals* of the form $s \neq t$, where s and t are terms. A literal that is not an inequality literal will be called a *regular literal*.

For a set S of literals or (possibly negated) predicates, $S^+ = \{a \mid a \in S\}$ and $S^- = \{a \mid not\ a \in S\}$. If S is a set of (possibly negated) predicates of arity n and $t_1, \ldots, t_n$ are terms, then $S(t_1, \ldots, t_n) = \{l(t_1, \ldots, t_n) \mid l \in S\}$. For a set S of atoms, $not\ S = \{not\ a \mid a \in S\}$.

A *program* is a finite set of rules $r : \alpha \leftarrow \beta$, where α is a finite set of regular literals and β is a finite set of literals. We denote as $head(r)$/$body(r)$ the set α/β, where α/β stands for a disjunction/conjunction.

Atoms, literals, rules, and programs that do not contain variables are *ground*. For a rule or a program R, let $vars(R)$, $preds(R)$, and $cts(R)$ be the sets of variables, predicates, and constants that occur in R, resp. A *universe* U for P is a non-empty countable superset of the constants in P: $U \supseteq cts(P)$. We call P_U the ground program obtained from P by substituting every variable in P by every element in U. Let $\mathcal{B}_P$ be the set of regular atoms that can be formed from a ground program P.

An *interpretation* I of a ground program P is a subset of $\mathcal{B}_P$. We write $I \models p(t_1, \ldots, t_n)$ if $p(t_1, \ldots, t_n) \in I$ and $I \models not\ p(t_1, \ldots, t_n)$ if $I \not\models p(t_1, \ldots, t_n)$. Also, for ground terms s, t, we write $I \models s \neq t$ if $s \neq t$. For a set of ground literals L, $I \models L$ if $I \models l$ for every $l \in L$. A ground rule $r : \alpha \leftarrow \beta$ is *satisfied* w.r.t. I, denoted $I \models r$, if $I \models l$ for some $l \in \alpha$ whenever $I \models \beta$. A ground constraint $\leftarrow \beta$ is satisfied w.r.t. I if $I \not\models \beta$.

For a positive ground program P, i.e., a program without not, an interpretation I of P is a *model* of P if I satisfies every rule in P; it is an *answer set* of P if it is a $\subseteq$- minimal model of P. When P is definite (does not contain disjunction) the minimal model of P can be computed using the well-known T_P operator: for a set of atoms B, let $T_P(B) = B \cup \{a \mid a \leftarrow \beta \in P \wedge B \models \beta\}$. Then, let $T_P^0(B) = B$ and $T_P^{i+1}(B) = T_P(T_P^i(B))$; the minimal model (answer set) of P, $\mathcal{M}(P)$, is defined as $\bigcup_{i=0}^{\infty} T_P^n(\emptyset)$. The derivation level of an atom a in $\mathcal{M}(P)$, $level(a, \mathcal{M}(P))$, is the least integer k such that $a \in T_P^k(\emptyset)$. For ground programs P containing not, the *GL-reduct* (Gelfond and Lifschitz 1988) w.r.t. I is defined as P^I, where P^I contains $\alpha^+ \leftarrow \beta^+$ for $\alpha \leftarrow \beta$ in P, $I \models not\ \beta^-$ and $I \models \alpha^-$. I is an *answer set* of a ground P if I is an answer set of P^I.

An *open interpretation* of a program P is a pair (U, M) where U is a universe for P and M is an interpretation of P_U. An *open answer set* of P is an open interpretation (U, M) of P, with M an answer set of P_U. For every atom $a \in M$, where (U, M) is an open answer set of P, $level(a, \mathcal{M}(P_U^M) = M)$ is finite (Heymans 2006).

Trees and forests: we introduce notation for trees and forests which extend those in (Vardi 1998). Let $\cdot$ be a concatenation operator between sequences of constants or natural numbers, $\mathbb{N}^+$ be the set of positive integers, and $\langle \mathbb{N}^+ \rangle$ be the set of all sequences of positive integers formed using the concatenation operator. We denote with ε the empty sequence: for every constant or natural number c, $c \cdot \varepsilon = c$. A *tree* T with root c, also denoted as T_c, is a set of nodes, where each node is a sequence of the form $c \cdot s$, where $s \in \langle \mathbb{N}^+ \rangle$, and for every $x \cdot d \in T_c$, $d \in \mathbb{N}^+$, it must be the case that $x \in T_c$. When the root of the tree is irrelevant, we will simply refer to the tree as T.

Given a tree T, its set of arcs is $A_T = \{(x, y) \mid x, y \in T, \exists n \in \mathbb{N}^+ . y = x \cdot n\}$. We denote with $succ_T(x) = \{y \in T \mid y = x \cdot i, i \in N^+\}$ the successors of a node x in T. For a node $y = x \cdot i \in T$, $prec_T(y) = x$.

A *forest* F is a set of trees $\{T_c \mid c \in C\}$, where C is a finite set of arbitrary constants. The set of nodes, N_F, and the set of arcs, A_F, of a forest F are defined as: $N_F = \cup_{T \in F} T$, and $A_F = \cup_{T \in F} A_T$, resp. For a node $x \in N_F$, let $succ_F(x) = succ_T(x)$, where $x \in T$ and $T \in F$. For a node $y = x \cdot i \in T$ and $T \in F$, $prec_F(y) = prec_T(y) = x$.

An *interconnected forest* EF is a tuple (F, ES), where $F = \{T_c \mid c \in C\}$ is a forest and $ES \subseteq N_F \times C$. The sets of nodes N_{EF} and arcs A_{EF} of an interconnected forest EF

are defined as: $N_{EF} = N_F$, and $A_{EF} = A_F \cup ES$, resp.

A Σ-labelled forest is a tuple (F, f) where F is an interconnected forest/tree and $f : N_F \to \Sigma$ is a labelling function, with Σ being a set of arbitrary symbols.

Forest Logic Programs

Forest Logic Programs (FoLPs) are a fragment of OASP which have the forest model property. They allow only for unary and binary predicates and tree-shaped rules. The tree-like structure of rules refers to the chaining pattern of rule variables: one variable can be seen as the root of a tree and the others as descendants such that for every arc in the tree, there is a positive binary literal in the body which connects the two corresponding variables. Inequalities between 'successor' variables can also appear in the body of such a rule; we will refer to the set of literals in the body of a rule formed only with the 'root' variable as the 'local part' and to the remaining part as the 'successor part'. FoLPs allow also for so-called 'free' rules, which are rules of the form: $p(\vec{t}) \vee not\ p(\vec{t}) \leftarrow$, where p is a unary/binary predicate and $\vec{t}$ is a unary/binary tuple of terms.

Definition 1 *A* forest logic program (FoLP) *is an open answer set program with only unary and binary predicates, and s. t. a rule is either:*

- *a* free rule*:*
$$a(s) \vee not\ a(s) \leftarrow , \tag{1}$$
or
$$f(s,t) \vee not\ f(s,t) \leftarrow \tag{2}$$
- *a unary rule:*
$$a(s) \leftarrow \beta(s), (\gamma_i(s,t_i), \delta_i(t_i))_{1 \leqslant i \leqslant m}, \psi \tag{3}$$
with $\psi \subseteq \bigcup_{1 \leqslant i \neq j \leqslant m} \{t_i \neq t_j\}$ *and* $m \in \mathbb{N}$*,*
- *or a* binary rule*:*
$$f(s,t) \leftarrow \beta(s), \gamma(s,t), \delta(t), \tag{4}$$
where in each rule above:
  - a *is a unary predicate, and* f *is a binary predicate,*
  - s*,* t*, and* $(t_i)_{1 \leqslant i \leqslant m}$ *are distinct terms,*
  - β*,* δ*, and* $(\delta_i)_{1 \leqslant i \leqslant m}$ *are sets of (possibly negated) unary predicates,*
  - γ*, and* $(\gamma_i)_{1 \leqslant i \leqslant m}$ *are sets of (possibly negated) binary predicates,*
  - *inequality does not appear in any* γ*:* $\neq \notin \gamma_m$*, for* $1 \leqslant m \leqslant k$*, and* $\neq \notin \gamma$*;*
  - *there is a positive atom that connects the head term* s *with any successor term which is a variable:* $\gamma_i^+ \neq \emptyset$*, if* t_i *is a variable, for every* $1 \leqslant i \leqslant m$*, and* $\gamma^+ \neq \emptyset$*, if* t *is a variable.*

A predicate q is *free* if it occurs in a free rule in P.

Example 1 *The following program* P *is a FoLP: it describes the fact that somebody who has read two different novels is a literature lover (unary rule* r_1*), a novel is something written by a novelist (unary rule* r_2*), and a novelist is somebody who wrote at least one novel (unary rule* r_3*). There are three free rules describing binary predicates* $read$*,* $writtenBy$*, and* $wrote$ *as being free. Finally, there are two facts (unary rules with empty bodies):* f_1 *and* f_2*.*

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

For a FoLP P, we denote with $upr(P)$, $bpr(P)$, $urul(P)$, and $brul(P)$, the sets of unary and binary predicates and unary and binary rules which occur in P, resp.

For a unary rule r of type (3), the degree of r, denoted as $deg(r)$, is the number k of successor variables which appear in the rule. Intuitively, it indicates the maximum number of successor individuals needed to make the body of the rule true. The degree of a free rule is 0. For a unary predicate p: $deg(p) = max\{deg(r) \mid p \in preds(head(r))\}$. Finally, the degree of a FoLP P is $deg(P) = \sum_{p \in upr(P)} deg(p)$. It over-approximates the number of successor individuals needed to satisfy all atoms of the form $p(x)$, where $p \in upr(P)$, for a given individual x.

Example 2 *Let* P *be the FoLP from Example 1. Then,* $deg(LitLover) = 2$*,* $deg(Novel) = 1$*,* $deg(Novelist) = 1$*, and thus,* $deg(P) = 4$*.*

Forest models: The forest model property of an OASP P is as follows: *if a unary predicate* p *is satisfiable, then there exists a model which satisfies* p *that can be seen as an interconnected forest*. The forest contains for each constant in P a tree having the constant as root, and possibly an additional tree with an anonymous root. The predicate checked to be satisfiable, p, belongs to the label of one of the root nodes.

Definition 2 *Let* P *be a program. A predicate* $p \in upr(P)$ *is* forest satisfiable *w.r.t.* P *if there exist an open answer set* (U, M) *of* P*; an interconnected forest* $EF = (\{T_\rho\} \cup \{T_a \mid a \in cts(P)\}, ES)$*, where* ρ *is a constant, possibly from* $cts(P)$*; and a labelling function* $ef : \{T_\rho\} \cup \{T_a \mid a \in cts(P)\} \cup A_{EF} \to 2^{preds(P)}$ *such that:*

- $p \in ef(\rho)$*,*
- $U = N_{EF}$*,*
- $ef(x) \in 2^{upr(P)}$*, when* $x \in T_\rho \cup \{T_a \mid a \in cts(P)\}$*,*
- $ef(x) \in 2^{bpr(P)}$*, when* $x \in A_{T_\rho}$*,*
- $M = \{p(x) \mid p \in ef(x), x \in N_{EF}\} \cup \{f(x,y) \mid f \in ef(x,y), (x,y) \in A_{EF}\}$*, and*
- *for every* $(z, z \cdot i) \in A_{EF}$*:* $ef(z, z \cdot i) \neq \emptyset$*.*

We call such a pair (U, M) *a* forest model*. A program* P *has the* forest model property*, if every unary predicate* p *that is satisfiable w.r.t.* P*, is forest satisfiable w.r.t.* P*.*

Proposition 1 *FoLPs have the forest model property.*

Example 3 *Consider again the FoLP* P *from Example 1. Figure 1 a) depicts a forest model which satisfies the unary predicate* $LitLover$*. Intuitively, the predicate is satisfied*

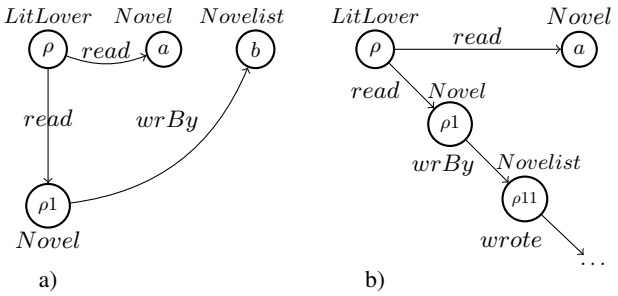

Figure 1: Forest model (a)/tentative forest model (b) for P

at the anonymous root ρ as there are two distinct $read$-successors of ρ where $Novel$ holds: a and $\rho 1$, and thus $LitLover(\rho)$ is 'supported' by a ground version of rule r_1. In turn, $Novel$ holds at $\rho 1$ as it is supported by rule r_2 grounded such that X is replaced with $\rho 1$ and Y is replaced with b. Note that every atom in the model is finitely supported: there is no infinite chain of dependencies of atoms in the model. This is a property of (open) answer sets also known as 'well-supportedness' (Fages 1991).

Consider in contrast the tentative model depicted in Figure 1 b). There, $Novel(\rho 1)$ is motivated by a $wrBy$-successor where $Novelist$ holds, which in turn is motivated by a $wrote$-successor where $Novel$ holds, etc. The interpretation is not a model, as $Novel(\rho 1)$ is not finitely motivated. One of the main challenges when designing algorithms to reason with FoLPs is ensuring that every atom in a model is well-supported.

As explained in the Introduction, FoLPs can be used to simulate reasoning with $\mathcal{SHOQ}$ ontologies. Thus, they can be viewed as an integrative formalism for reasoning with both ontologies and rules: (Feier and Heymans 2009) introduces so-called *f-hybrid knowledge bases (fKBs)* which are pairs of the form (Σ, P), where Σ is a $\mathcal{SHOQ}$ KB and P is a FoLP. The semantics of fKBs is defined in terms of pairs of interpretations, one for each component, which share the same domain and which agree on the interpretation of common symbols between the two components. The salient feature of f-hybrid KBs is that they impose no restriction on the occurrence of DL symbols in the FoLP KB.

A concept/unary predicate satisfiability preserving, polynomial, and modular translation Θ from $\mathcal{SHOQ}$ to FoLPs is provided in (Feier and Heymans 2009). The translation extends to fKBs as follows: given an fKB (Σ, P), its translation is simply $\Theta(\Sigma) \cup P$. Thus, any reasoning procedure for FoLPs can be employed to reason with fKBs. More details about the semantics, translation, and an extended example can be found in (Feier and Heymans 2013).

Fully Enriched Automata

Fully enriched automata (FEAs) were introduced in (Bonatti et al. 2008) as a tool to reason in hybrid graded μ-calculus. They accept forests as input. We describe them following (Bonatti et al. 2008).

For a set Y, we denote with $B^+(Y)$ the set of positive Boolean formulas over Y, where $true$ and $false$ are also allowed and where $\wedge$ has precedence over $\vee$. For a set $X \subseteq Y$ and a formula $\theta \in B^+(Y)$, we say that X satisfies θ iff assigning true to elements in X and assigning false to elements in $Y - X$ makes θ true. For $b > 0$, let $D_b = \{\langle 0\rangle, \langle 1\rangle, \ldots, \langle b\rangle\} \cup \{[0], [1], \ldots, [b]\} \cup \{-1, \varepsilon, \langle root\rangle, [root]\}$.

A fully enriched automaton (FEA) is a tuple $A = \langle \Sigma, b, Q, \delta, q_0, \mathcal{F}\rangle$, where Σ is a finite input alphabet, $b > 0$ is a counting bound, Q is a finite set of states, $\delta : Q \times \Sigma \to B^+(D_b \times Q)$ is a transition function, $q_0 \in Q$ is an initial state, and $\mathcal{F} = \{\mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_k\}$, where $\mathcal{F}_1 \subseteq \mathcal{F}_2 \subseteq \ldots \subseteq \mathcal{F}_k = Q$ is a *parity acceptance condition*. The number k of sets in $\mathcal{F}$ is the *index* of the automaton.

A *run* of a FEA on a labeled forest (F, V) is an $N_F \times Q$-labeled tree (T_c, r) such that:

- $r(c) = (d, q_0)$, for some root d in F, and
- for all $y \in T_c$ with $r(y) = (x, q)$ and $\delta(q, V(x)) = \theta$, there is a (possibly empty) set $S \subseteq D_b \times Q$ such that S satisfies θ and for all $(d, s) \in S$, the following hold:
  - if $d \in \{-1, \varepsilon\}$, then $x \cdot d$ is defined and there is $j \in \mathbb{N}^+$ such that $y \cdot j \in T_c$ and $r(y \cdot j) = (x \cdot d, s)$;
  - if $d = \langle n\rangle$, then there is a set $M \subseteq succ_F(x)$ of cardinality $n + 1$ such that for all $z \in M$, there is $j \in \mathbb{N}^+$ such that $y \cdot j \in T_c$ and $r(y \cdot j) = (z, s)$;
  - if $d = [n]$, then there is a set $M \subseteq succ_F(x)$ of cardinality n such that for all $z \in succ_F(x) - M$, there is $j \in \mathbb{N}^+$ such that $y \cdot j \in T_c$ and $r(y \cdot j) = (z, s)$;
  - if $d = \langle root\rangle$ ($d = [root]$), then for some (all) root(s) $c \in F$ there exists $j \in \mathbb{N}^+$ such that $y \cdot j \in T_c$ and $r(y \cdot j) = (c, s)$;

If θ above is true, then y does not need to have successors. Moreover, since no set S satisfies $\theta = false$, there cannot be any run that takes a transition with $\theta = false$. A run is *accepting* if each of its infinite paths π is accepting, that is if the minimum i for which $Inf(\pi) \cap \mathcal{F}_i \neq \emptyset$, where $Inf(\pi)$ is the set of states occurring infinitely often in π, is even. The automaton accepts a forest iff there exists an accepting run of the automaton on the forest. The language of A, denoted $\mathcal{L}(A)$, is the set of forests accepted by A. We say that A is non-empty if $\mathcal{L}(A) \neq \emptyset$.

Theorem 1 (Corollary 4.3 (Bonatti et al. 2008)) *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 $(b + 2)^{\mathcal{O}(n^3 \cdot k^2 \cdot \log k \cdot \log b^2)}$.*

From Forest Logic Programs to Fully Enriched Automata

In this section we encode satisfiability checking of unary predicates with respect to FoLPs as emptiness checking for FEAs.

We start by introducing for every FoLP P and unary predicate p a class of FEAs $A^{p,P}_{\rho,\theta}$, where ρ is one of $cts(P)$ or a new anonymous individual and $\theta : cts(P) \cup \{\rho\} \to$

$2^{upr(P)\cup cts(P)\cup\{\rho\}}$ is a function which has the following properties: $o_i \in \theta(o_i)$, and $o_j \notin \theta(o_i)$, for every $o_i, o_j \in cts(P) \cup \{\rho\}$, such that $o_i \neq o_j$. Furthermore, $p \in \theta(c)$, where c is one of $cts(P) \cup \{\rho\}$ and c is ρ if $\rho \notin cts(P)$. Intuitively $A^{p,P}_{\rho,\theta}$ will accept forest models of p with respect to P encoded in a certain fashion: in particular, for every root in the forest model, the root node will appear in its own label; function θ fixes a content for the label of each root of accepted forest models.

In the following, let $d = deg(P)$. We will also denote with PAT_P the set $\{*\} \cup cts(P)$ of *term patterns*, where $*$ stands for a generic anonymous individual: we say that a term t matches a term pattern pt, and write $t \mapsto pt$ iff $t = pt$, when t is a constant. In all other cases, the match trivially holds. Term patterns will be used in our encoding as a unification mechanism between terms occurring in a FoLP (variables and constants) and elements in a universe (anonymous individuals and constants): a variable will match with a constant or an anonymous individual, but a constant will match only with itself. The automata $A^{p,P}_{\rho,\theta}$ will run on forests labelled using the following alphabet: $\Sigma = 2^S$, where $S = upr(P) \cup \{1, \ldots, d\} \cup cts(P) \cup \{\rho\} \cup \{\uparrow^o_f |\ f \in bpr(P)\} \cup \{\downarrow^t_f |\ f \in bpr(P), t \in \text{PAT}_P\}$.

Unlike the case for forest models, the arcs of forests accepted by FEAs are not labelled: as such, binary predicates occur in the label of nodes in an adorned form. These adorned predicates are either of the form $\downarrow^t_f$, in which case they represent an f-link between the predecessor of the labelled node, which has term pattern t and the node itself, or of the form $\uparrow^o_f$, in which case the current node is linked to a constant o from P via the binary predicate f. Additionally, besides unary predicates, labels might contain natural numbers and constants, which will be used as an addressing mechanism for successors of a given node and nodes which stand for constants in accepted forests, resp. The set of states of the automaton are as follows: $Q = Q_i \cup Q_+ \cup Q_-$, with:

- $Q_i = \{q_0, q_1, q_o, q_{\neg k}\}$,
- $Q_+ = \{q_{t,a}, q_{t,r_a}, q_{t_1,t_2,u}, q_{t_1,t_2,r_f}, q_{k,t,*,u}\}$, and
- $Q_- = \{q_{\overline{t,a}}, q_{\overline{t,r_a}}, q_{\overline{t_1,t_2,u}}, q_{\overline{t_1,t_2,r_f}}, q_{\overline{k,t,*,u}}\}$,

where $o \in cts(P) \cup \{\rho\}$, $a \in upr(P)$, $f \in bpr(P)$, u is of the form a, f, $not\ a$ or $not\ f$, $k \in \{1, \ldots, d\}$, $r_a \in urul(P)$, $r_f \in brul(P)$, and $t, t_1, t_2 \in \text{PAT}_P$. We will refer to Q_+ and Q_- as the positive and the negative states of $A^{p,P}_{\rho,\theta}$, resp. Intuitively, positive states are used to motivate the presence of atoms in an open answer set while negative states are used to motivate the absence of atoms in an open answer set. Q_i contains some additional states, like q_0, the initial state, q_1, a state which will be visited recursively in every node of the forest, q_o, a state corresponding to the initial visit of constant nodes, and $q_{\neg k}$, a state which asserts that for every node in an accepted forest there must be at most one successor which has k in the label.

We describe in detail the transition function of $A^{p,P}_{\rho,\theta}$. The initial transition prescribes that the automaton visits a root of the forest in state q_o, for every $o \in cts(P) \cup \{\rho\}$:

$$\delta(q_0, \sigma) = \bigwedge_{o \in cts(P)\cup\{\rho\}} (\langle root \rangle, q_o) \tag{5}$$

In every such state q_o, it should hold that o and only o is part of the label. Furthermore, the automaton justifies the presence and absence of each unary predicate a and each adorned upward binary predicate in the label[1] by entering states $q_{o,a}$, $q_{o,o',f}$, $q_{\overline{o,a}}$, and $q_{\overline{o,o',f}}$ resp. At the same time every successor of the constant node is visited in state q_1. Let $c^o_P = cts(P) \cup \{\rho\} - \{o\}$. Then:

$$\begin{aligned}\delta(q_o, \sigma) = o \in \sigma \wedge \bigwedge_{o' \in c^o_P} o' \notin \sigma \wedge \bigwedge_{a \in \theta(o)} (\varepsilon, q_{o,a}) \wedge \bigwedge_{a \notin \theta(o)} \\ (\varepsilon, q_{\overline{o,a}}) \wedge \bigwedge_{\uparrow^{o'}_f \in \theta(o)} (\varepsilon, q_{o,o',f}) \wedge \bigwedge_{\uparrow^{o'}_f \notin \theta(o)} (\varepsilon, q_{\overline{o,o',f}}) \wedge ([0], q_1)\end{aligned} \tag{6}$$

Whenever the automaton finds itself in state q_1 it tries to motivate the presence and absence of each unary and adorned binary predicate in its label and then it recursively enters the same state into each successor of the current node. It also ensures that for each integer $1 \leqslant k \leqslant d$, the labels of each but one successor do not contain k:

$$\begin{aligned}\delta(q_1, \sigma) = ([0], q_1) \wedge \bigwedge_{1 \leqslant k \leqslant d} ([1], q_{\neg k}) \wedge \bigwedge_{a \in \sigma} (\varepsilon, q_{*,a}) \wedge \bigwedge_{a \notin \sigma} (\varepsilon, q_{\overline{*,a}}) \wedge \\ \bigwedge_{\downarrow^t_f \in \sigma} (\varepsilon, q_{t,*,f}) \wedge \bigwedge_{\downarrow^t_f \notin \sigma} (\varepsilon, q_{\overline{t,*,f}}) \wedge \bigwedge_{\uparrow^{o'}_f \in \sigma} (\varepsilon, q_{*,o',f}) \wedge \bigwedge_{\uparrow^{o'}_f \notin \sigma} (\varepsilon, q_{\overline{*,o',f}})\end{aligned} \tag{7}$$

$$\delta(q_{\neg k}, \sigma) = k \notin \sigma \tag{8}$$

To motivate the presence of a unary/binary predicate in the label of a node, the automaton checks whether the given predicate is free (we assume that $free(a)$ evaluates to true if a is free, and, to false, otherwise) or finds a unary/binary rule which supports the predicate. Note the distinction concerning the term pattern for the node where the predicate has to hold. In the case of unary predicates, if the node is a constant, there is first a preliminary check that we are at the right node - this is needed as later the automaton will visit all roots in this state. In the case of binary predicates, depending on the term pattern, the label is checked for different types of adorned binary atoms. In all cases, only rules whose head terms match the given term patterns are chosen to motivate the presence of predicates in the label.

$$\delta(q_{*,a}, \sigma) = a \in \sigma \wedge \Big(free(a) \vee \bigvee_{r_a : a(s) \leftarrow \beta} (\varepsilon, q_{*,r_a})\Big) \tag{9}$$

$$\delta(q_{o,a}, \sigma) = o \notin \sigma \vee a \in \theta(o) \wedge \Big(free(a) \vee \bigvee_{\substack{r_a : a(s) \leftarrow \beta, \\ s \mapsto o}} (\varepsilon, q_{o,r_a})\Big) \tag{10}$$

$$\delta(q_{t,*,f}, \sigma) = \downarrow^v_f \in \sigma \wedge \Big(free(f) \vee \bigvee_{\substack{r_f : f(s,v) \leftarrow \beta, \\ s \mapsto t, \\ v \mapsto *}} (\varepsilon, q_{v,*,r_f})\Big) \tag{11}$$

[1] As constants have no predecessors in the forest, there will be no adorned downward predicates in the label.

$$\delta(q_{t,o,f},\sigma) = \uparrow_f^o \in \sigma \wedge \Big(free(f) \vee \bigvee_{\substack{r_f : f(s,v) \leftarrow \beta, \\ s \mapsto t, \\ v \mapsto o}} (\varepsilon, q_{t,o,r_f}) \Big) \quad (12)$$

Let $r_a : a(s) \leftarrow \beta(s), (\gamma_i(s,v_i), \delta_i(v_i))_{1 \leqslant i \leqslant m}, \psi$ be a unary rule. Then, we denote with J_{r_a} a multiset $\{j_i \mid 1 \leqslant i \leqslant m, j_i \in \{1,\dots,d\} \cup cts(P)\}$ s. t.:

- 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 $j_i \neq j_l$.

Intuitively, a multiset provides a way to satisfy the successor part of a unary rule in a forest model by identifying the successor terms of the rule with either successors of the current element in the model or constants in the program. We next describe the transition function for unary rules:

$$\delta(q_{t,r_a},\sigma) = \bigwedge_{u \in \beta} (\varepsilon, q_{*,t,u}) \wedge \exists J_{r_a}. \Big(\bigwedge_{k=1}^{d} \bigwedge_{j_i = k} \bigwedge_{u \in \gamma_i \cup \delta_i} (\langle 0 \rangle, q_{k,t,*,u}) \wedge \bigwedge_{o \in cts(P)} \bigwedge_{j_i = o} \bigwedge_{u \in \gamma_i \cup \delta_i} (\varepsilon, q_{t,o,u}) \Big) \quad (13)$$

State $q_{k,t,*,u}$ enforces that the (possibly negated) unary or adorned binary predicate u is (is not) part of the label of the k-th successor of a given node; we thus define:

$$\delta(q_{k,t_1,t_2,u},\sigma) = k \in \sigma \wedge \bigwedge_{j \neq k} j \notin \sigma \wedge (\varepsilon, q_{t_1,t_2,u}) \quad (14)$$

The state $q_{t_1,t_2,u}$ can be seen as a multi-state with different transitions depending on its arguments (two transitions have already been introduced as rules (11) and (12) above): if $t_2 = *$, one has the justify the presence/absence of u in the label of the current node; otherwise, when $t_2 = o$, one has to justify it from the label of the root node corresponding to constant o: note that, as it is not possible to jump directly to a given root node in the forest, nor to enforce that there will be a single root node corresponding to each constant, in transition (17) we visit each root node in state $q_{o,a}$:

$$\delta(q_{t_1,t_2,u},\sigma) = \begin{cases} (\varepsilon, q_{*,a}), & \text{if } t_2 = * \text{ and } u = a \quad (15) \\ ([root], q_{o,a}), & \text{if } t_2 = o \text{ and } u = a \quad (16) \\ a \notin \sigma, & \text{if } t_2 = * \text{ and } u = not\ a \quad (17) \\ a \notin \theta(o), & \text{if } t_2 = o \text{ and } u = not\ a \quad (18) \\ \downarrow_f^t \notin \sigma, & \text{if } t_2 = * \text{ and } u = not\ f \quad (19) \\ \uparrow_f^o \notin \theta(o), & \text{if } t_2 = o \text{ and } u = not\ f \quad (20) \end{cases}$$

For binary rules: $r_f : f(s,v) \leftarrow \beta(s), \gamma(s,v), \delta(v)$, when v is grounded using an anonymous individual, the check involves looking to the predecessor node to see if the local part of the rule is satisfied. When v is grounded using a constant, the local part of the rule is checked at the current node and the successor part at the respective constant. Note that the first term pattern in the first conjunct in both rules (21) and (22) is irrelevant as β contains only unary predicates:

$$\delta(q_{t,*,r_f},\sigma) = \bigwedge_{u \in \beta} (-1, q_{*,t,u}) \wedge \bigwedge_{u \in \gamma \cup \delta} (\varepsilon, q_{t,*,u}) \quad (21)$$

$$\delta(q_{t,o,r_f},\sigma) = \bigwedge_{u \in \beta} (\varepsilon, q_{*,t,u}) \wedge \bigwedge_{u \in \gamma \cup \delta} (\varepsilon, q_{t,o,u}) \quad (22)$$

In the following, we describe the transitions of the automaton in the negative states, i.e. the states which are used to motivate the absence of certain unary/binary predicates in the labels of the forest. If one ignores the checks for the absence of unary/adorned binary predicates in the label of the current node, the transition rules can be seen as dual versions of the ones for the counterpart positive states.

$$\delta(q_{\overline{*,a}},\sigma) = a \notin \sigma \wedge \bigwedge_{r_a : a(s) \leftarrow \beta} (\varepsilon, q_{\overline{*,r_a}}) \quad (23)$$

$$\delta(q_{\overline{o,a}},\sigma) = o \notin \sigma \vee a \notin \theta(o) \wedge \bigwedge_{\substack{r_a : a(s) \leftarrow \beta, \\ s \mapsto o}} (\varepsilon, q_{\overline{o,r_a}}) \quad (24)$$

$$\delta(q_{\overline{t,*,f}},\sigma) = \downarrow_f^t \notin \sigma \wedge \bigwedge_{\substack{r_f : f(s,v) \leftarrow \beta, \\ s \mapsto t, \\ v \mapsto *}} (\varepsilon, q_{\overline{v,*,r_f}}) \quad (25)$$

$$\delta(q_{\overline{t,o,f}},\sigma) = \uparrow_f^o \notin \sigma \wedge \bigwedge_{\substack{r_f : f(s,v) \leftarrow \beta \\ s \mapsto t, \\ v \mapsto o}} (\varepsilon, q_{\overline{v,*,r_f}}) \quad (26)$$

$$\delta(q_{\overline{t,r_a}},\sigma) = \bigvee_{u \in \beta} (\varepsilon, q_{\overline{*,t,u}}) \vee \forall J_{r_a}. \bigvee_{k=1}^{d} \bigvee_{j_i = k} \bigvee_{u \in \gamma_i \cup \delta_i} ([0], q_{\overline{k,t,*,u}}) \vee \bigvee_{o \in cts(P)} \bigvee_{j_i = o} \bigvee_{u \in \gamma_i \cup \delta_i} (\varepsilon, q_{\overline{t,o,u}}) \quad (27)$$

$$\delta(q_{\overline{k,t,*,u}},\sigma) = k \notin \sigma \vee (\varepsilon, q_{\overline{t,*,u}}) \quad (28)$$

$$\delta(q_{\overline{t_1,t_2,u}},\sigma) = \begin{cases} (\varepsilon, q_{\overline{*,a}}), & \text{if } t_2 = * \text{ and } u = a \quad (29) \\ ([root], q_{\overline{o,a}}), & \text{if } t_2 = o \text{ and } u = a \quad (30) \\ a \in \sigma, & \text{if } t_2 = * \text{ and } u = not\ a \quad (31) \\ a \in \theta(o), & \text{if } t_2 = o \text{ and } u = not\ a \quad (32) \\ \downarrow_f^t \in \sigma, & \text{if } t_2 = * \text{ and } u = not\ f \quad (33) \\ \uparrow_f^o \in \theta(o), & \text{if } t_2 = o \text{ and } u = not\ f \quad (34) \end{cases}$$

$$\delta(q_{\overline{t,*,r_f}},\sigma) = \bigvee_{u \in \beta} (-1, q_{\overline{*,t,u}}) \vee \bigvee_{u \in \gamma \cup \delta} (\varepsilon, q_{\overline{t,*,u}}) \quad (35)$$

$$\delta(q_{\overline{t,o,r_f}},\sigma) = \bigvee_{u \in \beta} (\varepsilon, q_{\overline{*,t,u}}) \vee \bigvee_{u \in \gamma \cup \delta} (\varepsilon, q_{\overline{t,o,u}}) \quad (36)$$

Finally we specify the parity acceptance condition. The index of the automaton is 2, with $\mathcal{F}_1 = \{q_{t,a}, q_{t_1,t_2,f} \mid a \in upr(P), f \in bpr(P), t, t_1, t_2 \in \text{PAT}_P\}$ and $\mathcal{F}_2 = Q$. Intuitively, paths in a run of the automaton correspond to dependencies of literals in the accepted model and by disallowing the infinite occurrence on a path of states associated to atoms in the model we ensure that only well-supported models are accepted.

Theorem 2 *Let P be a FoLP and let p be a unary predicate symbol. Then, p is satisfiable w.r.t. P iff there exists an automaton $A^{p,P}_{\rho,\theta}$ such that $\mathcal{L}(A^{p,P}_{\rho,\theta}) \neq \emptyset$.*

Proof Sketch. ($\Rightarrow$): Assume p is satisfiable w.r.t. P. Then, by Prop. 1, p is satisfied by a forest model (U, M). Let (EF, ef), with $EF = (F, ES)$ be the labelled forest which induces (U, M), as in Definition 2. Then, let $l : \{T_c \mid c \in cts(P) \cup \{\rho\}\}$ be a labelling function such that for every $y \in \{T_c \mid c \in cts(P) \cup \{\rho\}\}$, $l(y)$ is the least set containing: *(i)* $ef(y)$, *(ii)* $\{\uparrow_f^o \mid f \in ef(y, o), o \in cts(P)\}$, *(iii)* $\{\downarrow_f^y \mid f \in ef(z, y), z = prec_F(y), z \notin cts(P)\}$, *(iv)* $\{\downarrow_f^z \mid f \in ef(z, y), z = prec_F(y), z \in cts(P)\}$, *(v)* $\{y\}$, when $y \in cts(P) \cup \rho$, and *(vi)* $\{i\}$ if $y = z \cdot i$.

We define an automaton $A^{p,P}_{\rho,\theta}$ which accepts (F, l). Let ρ be the same as its homonym in the considered forest model and let θ be such that $\theta(o) = l(o)$, for every $o \in cts(P) \cup \{\rho\}$. We construct a run (T_c, r) of $A^{p,P}_{\rho,\theta}$ on (F, l) by first setting $r(c) = (\phi, q_0)$, where ϕ is the root of some forest in F. Then, for each $o \in cts(P) \cup \{\rho\}$, we introduce a successor of c, $c \cdot i$, such that $r(c \cdot i) = (o, q_o)$.

We then proceed inductively with the construction by maintaining the following invariant, for each $w \in T_r$:

- $r(w) = (y, q_{*,a})$ implies $a \in l(y)$;
- $r(w) = (y, q_{o,a})$ implies $o \notin l(y)$ or $a \in \theta(o)$;
- $r(w) = (y, q_{t,r_a})$ implies $a \in l(y)$, a is not free, and there is a rule $s \in P_U^M$ derived from r_a such that $head(s) = a(y)$, $M \models body(s)$, and $max_{b \in body(s)}(level(b, M)) < level(a(y), M)$;
- $r(w) = (y, q_{t,*,u})$ implies $a \in l(y)$, if $u = a$; $a \notin l(y)$, if $u = not\ a$; $\downarrow_f^t \in l(y)$, if $u = f$; $\downarrow_f^t \notin l(y)$, if $u = not\ f$;
- $r(w) = (y, q_{t,o,u})$ implies $a \in l(o)$, if $u = a$; $a \notin l(o)$, if $u = not\ a$; $\uparrow_f^o \in l(y)$, if $u = f$; $\uparrow_f^t \notin l(y)$, if $u = not\ f$;
- $r(w) = (y, q_{k,t,*,u})$ implies $k \in l(y)$, and $j \notin l(y)$, for every $1 \leqslant j \neq k \leqslant d$; $a \in l(y)$, if $u = a$; $a \notin l(y)$, if $u = not\ a$; $\downarrow_f^t \in l(y)$, if $u = f$; $\downarrow_f^t \notin l(y)$, if $u = not\ f$;
- $r(w) = (y, q_{t,*,r_f})$ implies $\downarrow_f^t \in l(y)$, f is not free, and there is a rule $s \in P_U^M$ derived from r such that $head(s) = f(prec_F(y), y)$, $M \models body(s)$, and $max_{b \in body(s)}\ level(b, M) < level(f(prec_F(y), y), M)$;
- $r(w) = (y, q_{t,o,r_f})$ implies $\uparrow_f^o \in l(y)$, f is not free, and there is a rule $s \in P_U^M$ derived from r such that $head(s) = f(y, o)$, $M \models body(s)$, and $max_{b \in body(s)}\ level(b, M) < level(f(y, o), M)$;
- $r(w) = (y, q_{\overline{t,a}})$ implies $a \notin l(y)$;
- $r(w) = (y, q_{\overline{t,r_a}})$ implies $a \notin l(y)$;
- $r(w) = (y, q_{\overline{t,*,u}})$ implies $a \notin l(y)$, if $u = a$; $a \in l(y)$, if $u = not\ a$; $\downarrow_f^t \notin l(y)$, if $u = f$; $\downarrow_f^t \in l(y)$, if $u = not\ f$;
- $r(w) = (y, q_{\overline{t,o,f}})$ implies $a \notin l(o)$, if $u = a$; $a \in l(o)$, if $u = not\ a$; $\uparrow_f^o \notin l(y)$, if $u = f$; $\uparrow_f^t \in l(y)$, if $u = not\ f$;
- $r(w) = (y, q_{\overline{k,t,*,u}})$ implies $k \notin l(y)$ or $a \notin l(y)$, if $u = a$; $a \in l(y)$, if $u = not\ a$; $\downarrow_f^t \notin l(y)$, if $u = f$; $\downarrow_f^t \in l(y)$, if $u = not\ f$;
- $r(w) = (y, q_{\overline{t,*,f}})$ implies $\downarrow_f^t \notin l(y)$;
- $r(w) = (y, q_{\overline{t,o,r_f}})$ implies $\uparrow_f^o \notin l(y)$.

We show how the invariant is preserved in two cases of the construction:

- Assume $r(w) = (y, q_{t,a})$, for some $w \in T_r$. Then, from the IH: $a \in l(y)$, and $a(y) \in M$. Then, there must be a finite n such that $level(a(y), M) = n$ and some rule s in P_U^M such that $max_{b \in body(s)}(level(b, M)) = n - 1$ (from which $a(y)$ has been derived at iteration n). Let r_a be the rule in P from which s has been derived and let $w \cdot i$ be a successor of w in T_r such that $r(w \cdot i) = (y, q_{t,r_a})$. The invariant is preserved.
- Assume $r(w) = (y, q_{t,r_a})$, for some $w \in T_r$. Then, from the IH: $a \in l(y)$, a is not free, and there exists some rule s in P_U^M derived from r_a: $a(y) \leftarrow \beta^+(y) \cup \bigcup_{1 \leqslant i \leqslant m}(\gamma_i^+(y, z_i) \cup \delta_i^+(z_i))$ such that $M \models body(s)$ and $max_{b \in body(s)}(level(b, M)) < level(a(y), M)$. Note that $M \models \beta(y) \cup \bigcup_{i=1}^m (\gamma_i(y, z_i) \cup \delta_i(z_i))$. As (U, M) is a forest model, each z_m must be of the form $y \cdot k$, for some $1 \leqslant k \leqslant d$, or of the form $o \in cts(P)$. Let J_{r_a} be a multiset such that $j_i = z_i$, if $z_i \in cts(P)$ and $j_i = k$ if $z_i = y \cdot k$. Then, we introduce the following successors of w in T_c (denoted just by their label):
  - $(y, q_{*,t,u})$, for every $u \in \beta$: the invariant holds as $M \models \beta(y)$.
  - $(y \cdot k, q_{k,t,*,u})$, for every $1 \leqslant k \leqslant d$, i such that $j_i = k$, and $u \in \gamma_i \cup \delta_i$: the invariant holds as $M \models \gamma_i(y, y \cdot k) \cup \delta_i(y \cdot k)$. It is also the case that $k \in l(y \cdot k)$ and $j \notin l(y \cdot k)$, for every $1 \leqslant j \neq k\ leqd$.
  - $(y, q_{t,o,u})$, for every o and i such that $j_i = o$ and $u \in \gamma_i \cup \delta_i$: the invariant holds as $M \models \gamma_i(y, o) \cup \delta_i(o)$.

The invariant ensures the existence of a run. We next show that every run T_c constructed as above is accepting, i.e. on every path of T_c there are finitely many occurrences of states of the form $q_{t,a}$ or $q_{t_1,t_2,f}$. Assume that every element $w \in T_c$ for which $r(w) = (y, q_{t,a})$, $r(w) = (y, q_{t,*,f})$, or $r(w) = (y, q_{t,o,f})$ is annotated with $level(a(y), M)$, $level(f(prec(y), y), M)$, or $level(f(y, o), M)$, resp. From the invariant of the construction it follows that level annotations decrease along each path of T_c. But the level of every atom in an open answer set is finite. Thus, the number of level annotations, and consequently the number of occurrences of such states must be finite.

($\Leftarrow$): Assume that there exists an automaton $A^{p,P}_{\rho,\theta}$ such that $\mathcal{L}(A^{p,P}_{\rho,\theta}) \neq \emptyset$. Then, there exists a labelled forest (F', f') and an accepting run (T_c, r) on (F', f') such that $r(c) = (\phi, q_0)$, for some root ϕ in F'. Let F be the forest containing the nodes $y \in N_{F'}$ for which either (i) there exists some $w \in T_c$ such that $r(w) = (y, q_o)$ or (ii) $prec_{F'}(y) \in N_F$ and there exists some $w \in T_c$ such that $r(w) = (y, q_{k,t_1,t_2,u})$. Assume C is the set of roots in F. Then let $\eta : N_F \rightarrow N_F \cup cts(P) \cup \{\rho\}$ be as follows: $\eta(y) = \begin{cases} o_i, & \text{if } o_i \in l(y), \\ \eta(c) \cdot s, & \text{if } y = c \cdot s, \text{ for } c \in C \end{cases}$ and let l be the following labeling function for N_F: $l(y) =$

$\begin{cases}\theta(\eta(y)), & \text{if } \eta(y) \in cts(P)\\ f'(y), & \text{if } \eta(y) \in N_F - cts(P)\end{cases}$.

Also, let: $U = \{\eta(y) \mid y \in N_F\}$, $M = \{a(\eta(y)) \mid a \in l(y) \cap upr(P) \wedge y \in N_F\} \cup \{f(\eta(z), \eta(z)\cdot i) \mid \downarrow_f^* \in l(y) \wedge y = z \cdot i \wedge y \in N_F\} \cup \{f(\eta(y), o_i) \mid \uparrow_f^{o_i} \in l(y) \wedge y \in N_F\}$.

We show that (U, M) is a forest model of P by showing that:

(i) M is a model of P_U^M, i.e every rule in P_U^M is satisfied by M: we check that every rule in P_U^M is satisfied.

Let $r' : a(\eta(y)) \leftarrow \beta^+(\eta(y)), (\gamma_i^+(\eta(y), \eta(y_i)), \delta_i^+(\eta(y_i)))_{1 \leqslant i \leqslant m}$ be a rule in P_U^M derived from a unary rule $r : a(s) \leftarrow \beta(s), (\gamma_i(s, t_i), \delta_i(t_i))_{1 \leqslant i \leqslant m}, \psi$ in P. Let J_r be the multiset formed of elements j_i, $1 \leqslant i \leqslant m$, such that:

$$j_i = \begin{cases}\eta(y_i), & \text{if } \eta(y_i) \in cts(P),\\ k, & \text{if } \eta(y_i) = \eta(x) \cdot k.\end{cases}$$

Assume $M \models body(r')$, but $M \not\models a(\eta(y))$. Then, $a \notin l(y)$ and there must be some $w \in T_c$ such that either:

- $r(w) = (y, q_{\overline{x,a}})$. Then, for every rule $r_a : a(s) \leftarrow \beta \in P$, there must be a node $w_{r_a} \in T_c$ such that $r(w_{r_a}) = (y, q_{\overline{x,r_a}})$. This holds also for rule r. Then, one of the following holds:
  - $*$ there exists $w' \in T_c$ such that $r(w') = (y, q_{\overline{x,x,u}})$, for some $u \in \beta$: then, either $u = a$ and $(y, q_{\overline{*,a}}) \in T_c$ and thus $a \notin l(y)$ or $M \not\models a(\eta(y))$ – contradiction, or for some $u = not\ a$, $a \in l(y)$ or $M \models a(\eta(y))$ – contradiction.
  - $*$ for every multiset J_{r_a} as in transition rule (14) (including J_r above) and every $j_i \in J_{r_a}$ either:
    - $\cdot$ there exist $1 \leqslant k \leqslant d$, $1 \leqslant i \leqslant m$, and $u \in \gamma_i \cup \delta_i$ such that $j_i = k$ and for every $y \cdot g \in F$, there is a node $w_g \in T_c$ such that $r(w_g) = (y \cdot g, q_{\overline{k,t,x,u}})$: then, there is $w_k \in T_c$ such that $r(w_k) = (y \cdot k, q_{\overline{k,t,x,u}})$ and either (1) $k \notin l(y \cdot k)$ – contradiction, (2) $u = a$, $a \notin l(y \cdot k)$: $a(\eta(y \cdot k)) \notin M$ or $M \not\models a(\eta(y_i))$, and thus $M \not\models \delta_i(\eta(y_i))$ – contradiction, (3) $u = not\ a$, $a \in l(y \cdot k)$: $a(\eta(y \cdot k)) \in M$ or $M \models a(\eta(y_i))$, and thus $M \not\models \delta_i(\eta(y_i))$ – contradiction, (4) $u = f$, $\downarrow_f^t \notin l(y \cdot k)$: $f(\eta(y), \eta(y \cdot k)) \notin M$ or $M \not\models f(\eta(y), \eta(y_i))$, and thus $M \not\models \gamma_i(\eta(y), \eta(y_i))$ – contradiction, (5) for some $u = not\ f$, $\downarrow_f^t \in l(y \cdot k)$: $f(\eta(y), \eta(y \cdot k)) \in M$ or $M \models f(\eta(y), \eta(y_i))$, and thus $M \not\models \gamma_i(\eta(y), \eta(y_i))$ – contradiction.
    - $\cdot$ there exists $1 \leqslant i \leqslant m$ such that $j_i = o$, for some $o \in cts(P)$, a node $w_o \in T_c$, and $u \in \gamma_i \cup \delta_i$ such that $r(w_o) = (y, q_{\overline{t,o,u}})$: similar to above we reach a contradiction with the fact that $M \models \gamma_i(\eta(y), o) \cup \delta_i(o)$.
- $r(w) = (y, q_{\overline{o,a}})$ with $\eta(y) = o$. Then, $o \notin l(y) = \theta(o)$ – contradiction or $a \notin \theta(o)$ and for every rule $r_a : a(s) \leftarrow \beta \in P$, such that $s \mapsto o$ there must be a node $w_{r_a} \in T_c$ such that $r(w_{r_a}) = (y, q_{\overline{o,r_a}})$ – similar to above.

Thus, in each case we obtained a contradiction and $M \models a(\eta(y))$: every unary rule is satisfied by M. Similarly it can be shown that every binary rule is satisfied as well.

(i) M is minimal: from the fact that (T_c, r) is accepting, it follows that every path starting at a state in Q^+ must be finite. For every node $w \in T_c$, such that $r(w) = (y, q)$, for some $q \in Q^+$, let:

$$d(w) = \begin{cases}0, \text{ if } w \text{ has no successors in } T_c,\\ 1 + max_{w \cdot i \in T_c}(d(w \cdot i)), \text{ otherwise.}\end{cases}$$

We show by induction on $d(w)$ that:

- $r(w) = (y, q_{x,a})$ implies $a(\eta(y)) \in \mathcal{M}(P_U^M)$,
- $r(w) = (y, q_{o,a})$ implies $o \notin l(y)$ or $a(o) \in \mathcal{M}(P_U^M)$,
- $r(w) = (y, q_{t,r_a})$ implies P_U^M contains a rule $a(\eta(y)) \leftarrow \beta$ such that $\mathcal{M}(P_U^M) \models \beta$,
- $r(w) = (y, q_{t,x,r_f})$ implies $y = z \cdot i$, for some i, and P_U^M contains $f(\eta(z), \eta(z) \cdot i) \leftarrow \beta$ such that $\mathcal{M}(P_U^M) \models \beta$,
- $r(w) = (y, q_{t,o,r_f})$ implies P_U^M contains $f(\eta(y), o) \leftarrow \beta$ such that $\mathcal{M}(P_U^M) \models \beta$,
- $r(w) = (y, q_{k,t,x,u})$ implies $y = z \cdot i$, $k \in l(y)$, for every $j \neq k$: $j \notin l(y)$, and: $a(\eta(y)) \in \mathcal{M}(P_U^M)$, if $u = a$; $a(\eta(y)) \notin \mathcal{M}(P_U^M)$, if $u = not\ a$; $f(\eta(z), \eta(z) \cdot i) \in \mathcal{M}(P_U^M)$, if $u = f$; $f(\eta(z), \eta(z) \cdot i) \notin \mathcal{M}(P_U^M)$, if $u = not\ f$;
- $r(w) = (y, q_{t,x,u})$ implies $y = z \cdot i$, and: $a(\eta(y)) \in \mathcal{M}(P_U^M)$, if $u = a$; $a(\eta(y)) \notin \mathcal{M}(P_U^M)$, if $u = not\ a$; $f(\eta(z), \eta(z) \cdot i) \in \mathcal{M}(P_U^M)$, if $u = f$; $f(\eta(z), \eta(z) \cdot i) \notin \mathcal{M}(P_U^M)$, if $u = not\ f$;
- $r(w) = (y, q_{t,o,u})$ implies: $a(o) \in \mathcal{M}(P_U^M)$, if $u = a$; $a(o) \notin \mathcal{M}(P_U^M)$, if $u = not\ a$; $f(\eta(y), o) \in \mathcal{M}(P_U^M)$, if $u = f$; $f(\eta(y), o) \notin \mathcal{M}(P_U^M)$, if $u = not\ f$.

Induction base: assume w is a leaf in T_c and $r(w) = (y, q)$, for some $q \in Q^+$ ($d(w) = 0$). Then, one of the following holds:

- $r(w) = (y, q_{t,a})$ and a is free: in this case $a(\eta(y)) \in M$ and thus P_U^M will contain rule $a(\eta(y)) \leftarrow$. Thus, $a(\eta(y)) \in \mathcal{M}(P_U^M)$.
- $r(w) = (y, q_{t_1,t_2,f})$ and f is free: similar to above.
- $r(w) = (y, q_{t_1,t_2,u})$ and $u = not\ a$ or $u = not\ f$: from the fact that (T_r, t) is a run and transition rules (17-20), it follows that $a \notin l(y)$ or $\downarrow_f^{t_2}$ $(t_2 = *)$/ $\uparrow_f^o$ $(t_2 = o) \notin l(y)$. Then, the claim follows from the definition of M.

Induction step: consider a non-leaf node $w \in T_c$ ($d(w) > 0$). We will analyse the case when $r(w) = (y, q_{t,r_a})$: then, for every $u \in \beta$ there is a successor $w \cdot i_{\beta,u}$ of w such that $r(w \cdot i_{\beta,u}) = (y, q_{x,t,u})$ and there exists a multiset J_{r_a} and successors $w \cdot m_{k,i,u}$ and $w \cdot m_{o,i,u}$ for w such that:

- $r(w \cdot m_{k,i,u}) = (y \cdot z_{k,i,u}, q_{k,t,x,u})$, for all pairs (k, i) such that $j_i = k$ and $u \in \gamma_i \cup \delta_i$;

- $r(w \cdot m_{o,i,u}) = (y, q_{t,o,u})$, for all pairs (o, i) such that $j_i = o$ and $u \in \gamma_i \cup \delta_i$.

From the IH, it follows that:

- $\mathcal{M}(P_U^M) \models u(\eta(y))$, for every $u \in \beta$, and thus $\mathcal{M}(P_U^M) \models \beta(\eta(y))$ (*1),
- $\mathcal{M}(P_U^M) \models u(\eta(y), \eta(y) \cdot z_{k,i,u})$, for every $u \in \gamma_i$ and $\mathcal{M}(P_U^M) \models u(\eta(y) \cdot z_{k,i,u})$, for every $u \in \delta_i$ and $k \in l(\eta(y) \cdot z_{k,i,u})$: as there is a unique successor $y \cdot z_k$ of y such that $k \in l(y \cdot z_k)$, it follows that: $\mathcal{M}(P_U^M) \models u(\eta(y), \eta(y) \cdot z_k)$, for every $u \in \gamma_i$ and $\mathcal{M}(P_U^M) \models u(\eta(y) \cdot z_k)$, for every $u \in \delta_i$ and thus $\mathcal{M}(P_U^M) \models \gamma_i(\eta(y), \eta(y) \cdot z_k) \cup \delta_i(\eta(y) \cdot z_k)$ (*2),
- $\mathcal{M}(P_U^M) \models u(\eta(y), o)$, for every $u \in \gamma_i$ and $\mathcal{M}(P_U^M) \models u(o))$, for every $u \in \delta_i$, where $j_i = o$, and thus $\mathcal{M}(P_U^M) \models \gamma_i(\eta(y), o) \cup \delta_i(o)$ (*3).

From (*1)-(*3) and the fact that $j_i \neq j_l$ implies $z_{j_i} \neq z_{j_l}$, it follows that $\mathcal{M}(P_U^M) \models \beta(\eta(y)) \cup \bigcup_{1 \leqslant k \leqslant d}(\gamma_i(\eta(y), \eta(y) \cdot z_k) \cup \delta_i(\eta(y) \cdot z_k))_{j_i=k} \cup \bigcup_{o \in cts(P)}(\gamma_i(\eta(y), o) \cup \delta_i(o))_{j_i=o}, \psi$, which is the body of a grounding of r_a in P_U with head $a(\eta(y))$. Then, by applying the reduct one obtains that $\mathcal{M}(P_U^M) \models \beta^+(\eta(y)) \cup \bigcup_{1 \leqslant k \leqslant d}(\gamma_i^+(\eta(y), \eta(y) \cdot z_k) \cup \delta_i^+(\eta(y) \cdot z_k))_{j_i=k} \cup \bigcup_{o \in cts(P)}(\gamma_i^+(\eta(y), o) \cup \delta_i^+(o))_{j_i=o}$, which is the body of a rule with head $a(\eta(y))$ in P_U^M.

The other cases can be treated similarly.

Then, as $a \in l(y) \cap upr(P)$ implies that there exists a node $w \in T_r$ such that $r(w) = (y, q_{t,a})$, $\downarrow_f^t \in l(y)$ implies that there exists a node $w \in T_r$ such that $r(w) = (y, q_{t,x,f})$, and $\uparrow_f^o \in l(y)$ implies that there exists a node $w \in T_r$ such that $r(w) = (y, q_{t,o,f})$ (from transition rule (7)), it follows that $M \subseteq \mathcal{M}(P_U^M)$, and thus M is a minimal model of P_U^M. □

Theorem 3 *Satisfiability checking of unary predicates w.r.t. FoLPs is* EXPTIME*-complete.*

Proof Sketch. That the task is EXPTIME-hard follows from the fact that satisfiability checking of unary concepts w.r.t. $\mathcal{SHOQ}$ ontologies is EXPTIME-complete (Schild 1991); the latter has been reduced to satisfiability checking of unary predicates w.r.t. FoLPs in (Feier and Heymans 2013).

For the upper bound, we observe that for a given P and p, there is an exponential number of automata $A_{\rho,\theta}^{p,P}$ (as there are an exponential number of possible θ). Each such automaton can be constructed in exponential time, and emptiness checking for $A_{\rho,\theta}^{p,P}$ can be performed again in exponential time. The latter follows from Theorem 1 and the fact that the branching factor and the number of states of $A_{\rho,\theta}^{p,P}$ are polynomial in the size of P, while its index is constant (viz. 2). Then, from Theorem 2 it follows that satisfiability checking of unary predicates w.r.t. FoLPs is in EXPTIME. □

Discussion and Conclusion

We have described a reduction of satisfiability checking of unary predicates w.r.t. FoLPs to emptiness checking of FEAs. This enabled us to establish a tight complexity bound on this reasoning task for FoLPs. Other reasoning tasks like consistency checking of FoLPs and skeptical and brave entailment of ground atoms can be polynomially reduced to satisfiability checking of unary predicates (Heymans 2006); thus, the complexity result applies to those tasks as well. Also, by virtue of the translation from fKBs to FoLPs, the result applies to fKBs as well: satisfiability checking of unary predicates w.r.t. fKBs is EXPTIME-complete. Thus, reasoning with FoLP rules and $\mathcal{SHOQ}$ ontologies is not harder than reasoning with $\mathcal{SHOQ}$ ontologies themselves.

In the introduction, we mentioned the special form of forest model property as one of the reasons FoLPs cannot be straightforwardly dealt with using 2ATAs. An additional technical difficulty we had to overcome compared to both encodings using 2ATAs and other encodings using FEAs, is the fact that FEAs, unlike 2ATAs, cannot explicitly address successor nodes in runs: they are suitable for encoding graded modalities/number restrictions where the same property has to hold/not to hold at a given number of successors. In our encoding, in order to assert that different properties hold at different successors (as the structure of FoLP rules allows), we had to devise an addressing mechanism for successors. Furthermore, as terms occurring in any position in FoLP rules might be constants, our encoding also made use of a unification mechanism.

Finally, as our result shows that FoLPs have the same complexity as CoLPs, we plan to further investigate the extension of the deterministic AND/OR tableau algorithm for CoLPs (Feier 2014) to FoLPs. As explained in (Feier 2014), such an extension is far from trivial.

References

Bonatti, P. A.; Lutz, C.; Murano, A.; and Vardi, M. Y. 2008. The complexity of enriched μ-calculi. *Logical Methods in Computer Science* 4(3).

Calì, A.; Gottlob, G.; and Lukasiewicz, T. 2009. Datalog± : A unified approach to ontologies and integrity constraints. In *ICDT*, volume 9, 14–30.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2007. Answering regular path queries in expressive description logics: An automata-theoretic approach. In *Proc. AAAI*, 391–396.

Calvanese, D.; Eiter, T.; and Ortiz, M. 2009. Regular path queries in expressive description logics with nominals. *IJCAI* 714–720.

Calvanese, D.; Giacomo, G. D.; and Lenzerini, M. 2002. 2ATAs make DLs easy. In *Proc. DL*, 107–118.

Eiter, T., and Šimkus, M. 2009. Bidirectional answer set programs with function symbols. In *Proc. IJCAI*, 765–771.

Eiter, T.; Ianni, G.; Lukasiewicz, T.; Schindlauer, R.; and Tompits, H. 2008. Combining answer set programming with description logics for the semantic web. *AI* 172(12-13).

Fages, F. 1991. A new fix point semantics for generalized logic programs compared with the well-founded and the stable model semantics. *New Generation Computing* 9(4).

Feier, C., and Heymans, S. 2009. Hybrid reasoning with Forest Logic Programs. In *Proc. ESWC*, volume 5554, 338–352. Springer.

Feier, C., and Heymans, S. 2013. Reasoning with Forest Logic Programs and f-hybrid knowledge bases. *TPLP* 3(13):395–463.

Feier, C. 2012. Worst-case optimal reasoning with Forest Logic Programs. In *Proc. KR 2012*.

Feier, C. 2014. *Reasoning with Forest Logic Programs*. PhD thesis (http://tinyurl.com/pbtfsll), TU Wien.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proc. of ICLP'88*, 1070–1080.

Heymans, S.; Van Nieuwenborgh, D.; and Vermeir, D. 2006. Conceptual Logic Programs. *AMAI* 47(1–2):103–137.

Heymans, S.; Van Nieuwenborgh, D.; and Vermeir, D. 2008. Open Answer Set Programming with guarded programs. *Transactions on Computational Logic* 9(4):1–53.

Heymans, S. 2006. *Decidable Open Answer Set Programming*. PhD thesis, Vrije Universiteit Brussel.

Krötzsch, M.; Rudolph, S.; and Hitzler, P. 2008. Description logic rules. In *Proc. ECAI*, 80–84. IOS Press.

Magka, D.; Krötzsch, M.; and Horrocks, I. 2013. Computing stable models for nonmonotonic existential rules. In *IJCAI*.

Motik, B., and Rosati, R. 2010. Reconciling description logics and rules. *Journal of the ACM* 57(5):30:1–30:62.

Motik, B.; Sattler, U.; and Studer, R. 2005. Query answering for OWL-DL with rules. *Journal of Web Semantics* 3(1):41–60.

Rosati, R. 2006. DL+log: Tight integration of description logics and disjunctive Datalog. In *Proc. KR*, 68–78.

Rosati, R. 2008. On combining description logic ontologies and nonrecursive datalog rules. In *Proc. RR*.

Schild, K. 1991. A correspondence theory for terminological logics: Preliminary report. In *IJCAI*, 466–471.

Vardi, M. Y. 1998. Reasoning about the past with two-way automata. In *Proc. ICALP*, 628–641. Springer.