Hybrid Reasoning with Forest Logic Programs

Cristina Feier, Stijn Heymans

Knowledge-Based Systems Group, Vienna University of Technology

European Semantic Web Conference, 31 May–4 June 2009

Overview

- What are Forest Logic Programs (FoLPs)?
  - subset of Open Answer Set Programming (OASP)
- How can one reason with FoLPs?
  - tableau algorithm inspired from Description Logics
- What are FoLPs useful for?
  - integrating $\mathcal{SHOQ}$ KBs with (unsafe) FoLP rules: *f-hybrid knowledge bases*

Overview

- What are Forest Logic Programs (FoLPs)?
  - subset of Open Answer Set Programming (OASP)
- How can one reason with FoLPs?
  - tableau algorithm inspired from Description Logics
- What are FoLPs useful for?
  - integrating $\mathcal{SHOQ}$ KBs with (unsafe) FoLP rules: *f-hybrid knowledge bases*

Overview

- What are Forest Logic Programs (FoLPs)?
  - subset of Open Answer Set Programming (OASP)
- How can one reason with FoLPs?
  - tableau algorithm inspired from Description Logics
- What are FoLPs useful for?
  - integrating $\mathcal{SHOQ}$ KBs with (unsafe) FoLP rules: *f-hybrid knowledge bases*

Overview

- What are Forest Logic Programs (FoLPs)?
  - subset of Open Answer Set Programming (OASP)
- How can one reason with FoLPs?
  - tableau algorithm inspired from Description Logics
- What are FoLPs useful for?
  - integrating $\mathcal{SHOQ}$ KBs with (unsafe) FoLP rules: *f-hybrid knowledge bases*

Overview

- What are Forest Logic Programs (FoLPs)?
  - subset of Open Answer Set Programming (OASP)
- How can one reason with FoLPs?
  - tableau algorithm inspired from Description Logics
- What are FoLPs useful for?
  - integrating $\mathcal{SHOQ}$ KBs with (unsafe) FoLP rules: *f-hybrid knowledge bases*

Overview

- What are Forest Logic Programs (FoLPs)?
  - subset of Open Answer Set Programming (OASP)
- How can one reason with FoLPs?
  - tableau algorithm inspired from Description Logics
- What are FoLPs useful for?
  - integrating $\mathcal{SHOQ}$ KBs with (unsafe) FoLP rules: *f-hybrid knowledge bases*

Part I

Forest Logic Programs

Open Answer Set Programming

Syntax: same as Answer Set Programming without function symbols

Semantics: interpretations are defined with respect to *open* domains

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

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

Open Answer Set Programming - Example

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

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

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

- $(\{john, x_1, x_2, \ldots\}, \{pass(john), fail(x_1), fail(x_2), \ldots\})$,

Forest Logic Programs - subset of OASP

OASP is undecidable: shown by reduction from undecidable *domino problem*.

Syntax restrictions:

- tree-shaped rules:
  - only unary and binary literals are allowed
  - unary literals correspond to nodes, binary to arcs
  - no constants: Conceptual Logic Programs - tree model property (decidable)
  - constants allowed: **Forest Logic Programs!** - forest model property (assumed to be decidable)
- guarded fragment

FoLP Rules

Free Rules:
$a(s) \vee not\ a(s) \leftarrow$ or $f(s,t) \vee not\ f(s,t) \leftarrow$

Unary Rules:
$r : a(s) \leftarrow \beta(s), (\gamma_m(s,t_m), \delta_m(t_m))_{1 \leq m \leq k}, \psi$, where

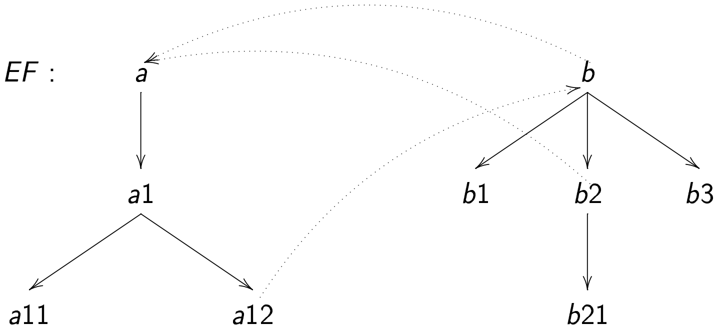
1. $\psi \subseteq \bigcup_{1 \leq i \neq j \leq k} \{t_i \neq t_j\}$ and $\{\neq\} \cap \gamma_m = \emptyset$ for $1 \leq m \leq k$,
2. $\forall t_i \in vars(r) : \gamma_i^+ \neq \emptyset$

Binary Rules: $f(s,t) \leftarrow \beta(s), \gamma(s,t), \delta(t)$ with $\{\neq\} \cap \gamma = \emptyset$ and $\gamma^+ \neq \emptyset$

Constraints: $\leftarrow a(s)$ or $\leftarrow f(s,t)$

Extended Forest

An extended forest is a tuple (F, ES) where F is a forest (set of trees) and ES is a set containing some extra arcs from any node in a tree in F to some root of a tree in F. We denote with N_{EF} the nodes of EF and with A_{EF} its arcs (including ES).



Forest Model Property

If a unary predicate p is satisfiable w.r.t. a FoLP P then p is forest satisfiable w.r.t. P.

A unary predicate p is *forest satisfiable* w.r.t. a FoLP P if there is an open answer set (U, M) of P, an extended forest $EF = (F, ES)$, and a labeling function $\mathcal{L} : N_{EF} \cup A_{EF} \to 2^{preds(P)}$ such that:

- F is a set of trees with roots from $\{\varepsilon\} \cup cts(P)$, one for each member of the set, where $\varepsilon \in cts(P) \cup \{x\}$
- $U = N_{EF}$
- $\mathcal{L}(x) \in 2^{upreds(P)}$, if $x \in N_{EF}$ and $\mathcal{L}(x) \in 2^{bpreds(P)}$, if $x \in A_{EF}$
- $p \in \mathcal{L}(\varepsilon)$
- $q(x) \in M$ iff $q \in \mathcal{L}(x)$ and $x \in N_{EF} \cup A_{EF}$
- $\mathcal{L}(x) \neq \emptyset$, for $x \in A_{EF}$

Forest Model Property Example

Consider the open answer set $OA = (\{x, a, z, y\}, \{p(x), g(x, z), q(z), f(z, a), q(a), f(a, y)\})$ for a FoLP P. p is forest-satisfiable w.r.t. P:

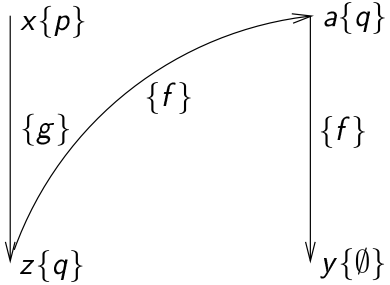


Figure: A forest model

Completion Structure for a FoLP

A completion structure for a FoLP P is a tuple: $\langle EF, \textsc{ct}, G, \textsc{st} \rangle$

- EF is an extended forest - the universe
- $\textsc{ct} : N_{EF} \cup A_{EF} \rightarrow 2^{preds(P) \cup not\ (preds(P))}$: maps a node to a set of (possibly negated) unary predicates and an arc to a set of (possibly negated) binary predicates
- $G = \langle V, A \rangle$ is a directed graph with vertices $V \subseteq \mathcal{B}_{P_{N_{EF}}}$ and arcss $A \subseteq V \times V$
- $\textsc{st}$ is function which indicates which predicates in a node/arc are already expanded at a certain time in the computation process

Initial Completion Structure

An *initial completion structure* for checking satisfiability of a unary predicate p w.r.t. a FoLP P is a completion structure $\langle EF, G, \textsc{ct}, \textsc{st} \rangle$ with:

- $EF = (F, ES)$, F is a set of single-node trees with roots from $\{\varepsilon\} \cup cts(P)$, one for each member of the set, where $\varepsilon \in cts(P) \cup \{x\}$; $ES = \emptyset$
- $\textsc{ct}(\varepsilon) = \{p\}$
- G has one vertex $p(\varepsilon)$ and no arcs
- p in ε is unexpanded

$$\bullet x\{p^{unexp}\} \quad \bullet a \quad \bullet b \quad \bullet c$$

Figure: Initial completion structure for p w.r.t. P which has the constants a, b, and c

Expansion Rules

Expand unary/binary positive: motivates the presence of an atom $p(x)/f(x,y)$ in the open answer set

- a rule whose head matches $p(x)/f(x,y)$ is randomly picked up
- the rule is grounded such that the head variable(s) coincide with the current node/arc
- the completion structure is updated accordingly

Expansion Rules

Expand unary/binary negative: motivates the absence of an atom $p(x)/f(x,y)$ in the open answer set

- the body of every ground version of a rule whose head matches with $p(x)/f(x,y)$ has to be refuted
- all combinations of successor nodes have to be considered
- the rule is visited multiple times to check for new successors (unless all positive predicates are expanded and every predicate appears either in a positive or a negated form in the current node/arc)

Expansion Rules

Choose unary/binary: expand the partial model to a complete model

- randomly choose a unary/binary predicate p which does not appear in the current node/arc x and insert p or *not* p in $\textsc{ct}(x)$

Applicability Rules

Saturation: no expansion in the successor until the predecessor is saturated

Blocking: A node $x \in N_{EF}$ is *blocked* if it has a tree ancestor y s.t. $\textsc{ct}(x) \subseteq \textsc{ct}(y)$ and the set $paths_G(x, y) = \{(p, q) \mid (p(x), q(y)) \in paths_G\}$ is empty.

Redundancy: A node $x \in N_{EF}$ is *redundant* if it is not blocked, it is saturated and there are k (tree) ancestors of x, $(y_i)_{1 \leq i \leq k}$, where $k = 2^p(2^{p^2} - 1) + 3$, and $p = |upreds(P)|$, s. t. $\textsc{ct}(x) = \textsc{ct}(y_i)$ for every $1 \leq i \leq n$

Termination

Complete completion structure for a FoLP P: no expansion rules can be further applied

Clash-free complete completion structure for a FoLP P: a complete completion structure $CS = \langle EF, G, \text{CT}, \text{ST} \rangle$ for which: (1) CS is not contradictory; (2) EF does not contain redundant nodes; (3) G does not contain cycles.

A complete completion structure can be constructed by a finite number of applications of the expansion rules to an initial completion structure considering the applicability rules.

Soundness, Completeness, and Complexity

If there is a clash-free complete completion structure for p w.r.t. P then p is satisfiable w.r.t. P.

There is a clash-free complete completion structure for p w.r.t. P if p is satisfiable w.r.t. P.

The algorithm runs in 2-nexptime

Part III

F-Hybrid Knowledge Bases

F-Hybrid Knowledge Bases - Syntax

An *f-hybrid knowledge base* is a pair $\langle \Sigma, P \rangle$ where Σ is a $\mathcal{SHOQ}$ knowledge base and P is a FoLP.

- *DL predicates*: predicates in P which are also atomic concept or role names from Σ
- no predicates from P coincide with complex concept or role descriptions from Σ
- no Datalog safeness or *(weakly) DL safeness* is imposed for the rule component

F-Hybrid Knowledge Bases - Semantics

the *projection* $\Pi(P, \mathcal{I})$ of a ground FoLP P with respect to a given DL interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$:
for every rule r in P,

- if there exists a DL literal in the head of the form
  - $A(t_1, \ldots, t_n)$ with $(t_1, \ldots, t_n) \in A^{\mathcal{I}}$, or
  - *not* $A(t_1, \ldots, t_n)$ with $(t_1, \ldots, t_n) \notin A^{\mathcal{I}}$,

  then delete r,
- if there exists a DL literal in the body of the form
  - $A(t_1, \ldots, t_n)$ with $(t_1, \ldots, t_n) \notin A^{\mathcal{I}}$, or
  - *not* $A(t_1, \ldots, t_n)$ with $(t_1, \ldots, t_n) \in A^{\mathcal{I}}$,

  then delete r,
- otherwise, delete all DL literals from r.

F-Hybrid Knowledge Bases - Semantics

$(U, \mathcal{I}, M)$ is an *interpretation* of an f-hybrid knowledge base $\langle \Sigma, P \rangle$ if:

- U is a universe for P,
- $\mathcal{I} = (U, \cdot^{\mathcal{I}})$ is an interpretation of Σ, and
- M is an interpretation of $\Pi(P_U, \mathcal{I})$.

$(U, \mathcal{I}, M)$ is a *model* of $\langle \Sigma, P \rangle$ if $\mathcal{I}$ is a model of Σ and M is an answer set of $\Pi(P_U, \mathcal{I})$

F-Hybrid Knowledge Bases - Reasoning

Satisfiability checking w.r.t. f-hybrid knowledge bases can be reduced to satisfiability checking of FoLPs only:

- for each concept expression one introduces a new predicate together with rules that define the semantics of the corresponding DL construct.
- constraints encode the inclusion axioms
- the first-order interpretation of DL concept expressions is simulated using free rules.

There is a polynomial, non-modular, and faithful translation w.r.t. predicate satisfiability from $\mathcal{SHOQ}$ knowledge bases to FoLPs.

Satisfiability checking w.r.t. f-hybrid knowledge bases is in 2-nexptime.

Related and Future Work

Related Work:

- *R-hybrid KBs*: DL knowledge base and a disjunctive Datalog program where each rule is *weakly DL-safe*
- *Description Logic Rules* : decidable fragments of SWRL. Tree-shaped rules similar to the structure of FoLPs, but the semantics is a first-order one and not a minimal one
- $\mathbb{FDNC}$: an extension of ASP with function symbols where rules are syntactically restricted in order to maintain decidability. The restriction is somehow similar to the one for FoLPs, but $\mathbb{FDNC}$ rules are required to be safe

Future Work:

- extension of f-hybrid KBs bases and its reasoning algorithm, from $\mathcal{SHOQ}$ towards $\mathcal{SROIQ}$
- prototype implementation and optimization

Conclusions

- Reasoning support for a language which allows an innovative combination of ontologies and rules (no safeness condition is needed).
- Tableau algorithm for a non-monotonic yet not Herbrand-restricted formalism
- Decidability result for FoLPs