From Datalog Reasoning to Modular Structure of an Ontology

Changlong Wang[1,2], Zhiyong Feng[1] ,Guozheng Rao[1], Xin Wang[1], Xiaowang Zhang[1]

[1]School of Computer Science and Technology, Tianjin University,Tianjin 300072, China,
[2]School of Computer Science and Engineer Technology, NWNU,Lanzhou 730070, China,

Abstract. *We propose a novel approach to modularly structuring an ontology. First, a ontology TBox is transformed into a datalog program. Second, a tree, which represents forward-chaining proof and contains information about modules, is mapped to a digraph, which allows us to extract intended ontology module via breadth-first search. Finally, we develop an algorithm to compute the modular structure of an ontology and take an example to illustrate how our strategy might work in practice.*

Keywords: Ontology; Modular structure; Datalog; Digraph

1 Introduction

Ontologies, often treated as monolithic objects, suffer from an inherent lack of structure [1]. To address this problem, atomic decomposition(AD) [2] is proposed to represent the modular structure of an ontology. This approach exploits locality-based module [3] to induce a modular structure and allows users to explore the entire ontology in a sensible manner so that we can find appropriate module. Atomic decomposition, however, is intimately tied to locality-based modules and thus suffers from their limitations [1]: it is bound by axiom structure and the extracted modules are often big in size. The recently proposed module extraction technique [4], in which module extraction is reduced to reasoning problem in datalog, allows us to extract smaller modules. Furthermore, those trees used in representing forward-chaining proofs in datalog can bring some information about ontology structure. Those distinguishing features stimulate us to further observe modular structure of an ontology by applying datalog reasoning.

2 Preliminaries

Ontologies and Datalog Reasoning The ontology axioms are formalised as rules: function-free sentences of the form $\forall x.[\varphi(x) \rightarrow \exists y.[\bigvee_{i=1}^{i=n} \psi_i(x, y)]]$, where φ,ψ are conjunctions of distinct atoms. A signature Σ is a set of predicates and $Sig(F)$ denotes the signature of a formula F. A TBox TB is a finite set of rules. A fact γ is a function-free ground atom. An ABox AB is a finite set of facts. An ABox with predicates from the signature Σ is denoted by Σ-ABox. Given a datalog program P and ABox AB, the materialisation of $P \bigcup AB$ can be computed by forward-chaining. A proof of γ in $P \bigcup AB$ is pair $\rho = (T, \lambda)$ where T is a tree, λ is a mapping from nodes in T to facts, and from edges in T to rules in P, such that for each node v the followings hold: (1) $\lambda(v) = \gamma$ if v is the root of T; (2) $\lambda(v) \in AB$ if v is a leaf; and (3) if v has children $w_1, ..., w_n$ then each edge from v to w_i is labelled by the same rule r and $\lambda(v)$ is a consequence of r and $\lambda(w_1), ..., \lambda(w_n)$. In this paper, we call T a *proof tree*.

Module Extraction via Datalog Reasoning The module extraction approach proposed in [4] consists of three steps: (1) Pick a substitution θ mapping all existentially quantified variables in TB to constants and then transform TB into a datalog program P. (2) Pick a Σ-ABox AB_0 and materialise $P \bigcup AB_0$. (3) Pick a set AB_r of "relevant facts" in the materialisation and compute the supporting rules in P for each such fact. Consequently, the module M consists of all rules in TB that yield some supporting rule in P and is fully determined by the substitution θ and the ABox AB_0 and AB_r. Intuitively, AB_0 determines the module "topic" and AB_r determines which rules should be contained in M. Formally, this approach is defined as follows:

Definition 1 [4]. A module setting for TB and Σ is a tuple $\chi = (\theta, AB_0, AB_r)$ with θ a substitution from existentially quantified variables in TB to constants, AB_0 a Σ-ABox, AB_r a $Sig(TB)$-ABox, and such that no constant in χ occurs in TB.

The program of χ is the smallest program P^χ containing, for each $r = \varphi(x) \rightarrow \exists y.[\bigvee_{i=1}^{i=n} \psi_i(x, y)]$ in TB, the rule $\varphi \rightarrow \bot$ if $n = 0$ and all rules $\varphi \rightarrow \gamma\theta$ for each $1 \leq i \leq n$ and each atom γ in ψ_i. The support of χ is the set of rules $r \in P^\chi$ that support a fact from AB_r in $P^\chi \cup AB_0$. The module M^χ of χ is the set of rules in TB that have a corresponding datalog rule in the support of χ. $\Diamond$

Definition 1 shows that there exists some relationship between rules in P^χ which make it feasible in characterizing the modular structure of an ontology by datalog reasoning:

Proposition 1. Given two datalog rules dr_1 and dr_2 in a proof tree T, if the the head of dr_2 is the body of dr_1, then, for ever module M^χ, $dr_2 \in M^\chi$ implies $dr_1 \in M^\chi$

3 From Proof Tree to Modular Structure of an Ontology

For the materialisation of AB_0 and P, we need to construct proof trees for each fact entailed by $AB_0 \bigcup P$. Consequently, the module M^χ for $\chi = (\theta, AB_0, AB_r)$ has one or more corresponding proof trees.

Definition 2. Given $\chi = (\theta, AB_0, AB_r)$, a module M^χ is called *single tree module* (STM) if it has one corresponding proof tree, A module is called *multiple trees module* (MTM) if it has more than one corresponding proof trees.

In this paper, we mainly discuss STM. We use T_M to denote the proof tree T for a module M, and use M_T to denote the module M generated from a proof tree T. Clearly, a module M^χ for $\chi = (\theta, AB_0, AB_r)$ is STM, if there exists only one fact $\gamma \in AB_r$ whose support contains supporting rules of any other fact $\beta \in AB_r$.

Definition 3 [2]. A module M is called *fake* if there exist two incomparable (w.r.t. set inclusion) modules $M_1 \neq M_2$, such that $M_1 \bigcup M_2 = M$; a module is called *genuine* if it is not fake.

Proposition 2. A STM is a genuine module.

Definition 4. Let T be a tree, a subtree ST in T is called breadth-first search tree (BFST), if ST is generated by breadth-first search starting from the root in T.

Theorem 1. Let T_{M^χ} be a proof tree for the genuine module M^χ and ST be a BFST in T_{M^χ}, the set of rules in TB that have a corresponding datalog rule in ST is a $\chi-$module for $\chi = (\theta, AB_0^{'}, AB_r^{'})$, where $AB_0^{'}$ consists of the facts that have corresponding leaves in ST and $AB_r^{'}$ consists of the single fact that corresponds to the root of ST.

Proof. From Definition 1 and the procedure of module extraction in [4], $AB_0^{'}$ can be picked as the initial ABox AB_0, and $AB_r^{'}$ as the relevant fact set AB_r, the only fact in AB_r is the root of ST and its support contains supporting rules of other inner n-ode(corresponding to a fact in $P^\chi \bigcup AB_0^{'}$) in ST . Hence, the set of rules in TB that have a corresponding datalog rule in ST is a $\chi-$module. □

Theorem 1 shows that a module can be extracted from a proof tree T by breadth-first search. In order to represent the modular structure of original ontology, we map each proof tree T to a digraph DG by: (1) If rule r in TB has a corresponding datalog rule dr in T, DG contains node r; (2) For two node r_1 and r_2 in DG and their corresponding datalog rules dr_1 and dr_2 in T, if the body of dr_1 is the head of dr_2, there exist a edge from node r_1 to r_2 in DG.

The mapping from datalog proof tree to digraph has no impact on the dependent relationship between rules. In other words, we can obtain the same module from datalog proof tree and its corresponding digraph. Those obtained digraphs exactly capture the modular structure of an ontology.

4 Computing the Modular Structure of an Ontology

From previous discussion, it is feasible to design an algorithm to compute the modular structure of an ontology TBox TB. In the algorithm, we need construct some proof trees, such that each rule in TB have a corresponding datalog rule in those trees. Algorithm 1 outlines our approach to computing the modular structure of an ontology. In Algorithm 1, we can pick a general substitution θ and transform TB into a datalog program P, and pick $Sig(TB)$-ABox as the initial ABox. From Definition 2, Definition 4, and Definition 8 in [4], the modular structure in this paper can be induced on Σ-implication inseparable, Σ-fact inseparable, and Σ-model inseparable module.

To conclude, we illustrate our approach by an ontology TBox TB containing 6 axioms: $\alpha_1 : A \sqsubseteq \exists R.B$, $\alpha_2 : A \sqsubseteq \exists R.C$, $\alpha_3 : B \sqcap C \sqsubseteq D$, $\alpha_4 : D \sqsubseteq \exists S.E$, $\alpha_5 : D \sqsubseteq \forall S.F$, $\alpha_6 : \exists S(E \sqcap F) \sqsubseteq G$. Those axioms are formalised as rules as follows, where the three rules r_1,r_2, and r_4 have two corresponding datalog rules, respectively. The mapping from proof trees to modular structure is shown in Figure 1.

Algorithm 1 Computing modular structure of an ontology

1: Input: a TBox TB
2: Output: the modular structure of TB — a set of digraphs
3: map TB to a datalog program P
4: use a $Sig(TB)$-ABox as AB_0 and build proof trees for each fact in the materialisation of $AB_0 \bigcup P$
5: map each proof tree to a digraph DG;
6: return the set of DG

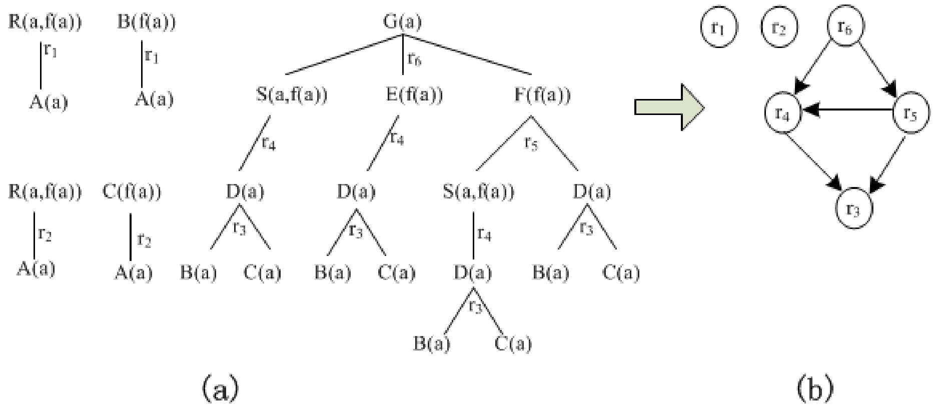

Fig. 1. (a) proof tree (b) modular structure

$$
\begin{array}{llll}
r_1 : A(x) \to \exists y_1[R(x, y_1) \wedge B(y_1) & \Rightarrow & dr_1^{'} : A(x) \to R(x, f(x)), & dr_1^{''} : A(x) \to B(c) \\
r_2 : A(x) \to \exists y_2[R(x, y_2) \wedge C(y_2) & \Rightarrow & dr_2^{'} : A(x) \to R(x, f(x)), & dr_2^{''} : A(x) \to C(c) \\
r_3 : B(x) \wedge C(x) \to D(x) & & & \\
r_4 : D(x) \to \exists y_3[S(x, y_3) \wedge E(y_3) & \Rightarrow & dr_4^{'} : D(x) \to S(x, f(x)), & dr_4^{''} : D(x) \to E(c) \\
r_5 : D(x) \wedge S(x, y) \to F(y) & & & \\
r_6 : S(x, y) \wedge E(y) \wedge F(y) \to G(x) & & &
\end{array}
$$

5 Conclusion and Future Work

We have proposed an approach to modular structure of ontology by mapping datalog proof tree to digraph. In this modular structure, ontology module can be computed by the standard algorithm of breadth-first search. Our current work is preliminary, we will implement the proposed approach and evaluate it on real ontologies in the future.

Acknowledgments. This work is supported by the NSFC (61100049, 61373165) and the 863 Program of China (2013AA013204)

References

1. Chiara Del Vescovo.: The Modular Structure of an Ontology: Atomic Decomposition and its applications. PhD thesis, The University of Manchester, 2013.
2. Chiara Del Vescovo, Bijan Parsia, Ulrike Sattler, Thomas Schneider: The modular structure of an ontology: Atomic decomposition. In IJCAI, 2232-2237, 2011.
3. Bernardo Cuenca Grau, Ian Horrocks, Yevgeny Kazakov, Ulrike Sattler: Modular reuse of ontologies: Theory and practice. J. of Artificial Intelligence Research. 31, 273-318, 2008.
4. Ana Armas Romero, Mark Kaminski,Bernardo Cuenca Grau, Ian Horrocks: Ontology Module Extraction via Datalog Reasoning. In AAAI, 1410-1416, 2015.