1F1-02

Faster Ontology Reasoning with Typed Propositionalization

Maxime Clement   Ryutaro Ichise

National Institute of Informatics, Tokyo, Japan

Ontology reasoning is an important but complex problem allowing the discovery of new knowledge and the query and maintenance of complex data. For decision making in real-time systems, faster approach such as decision tables are often used. In this paper, we propose a technique to perform efficient ontology reasoning with decision tables. Our experiments show a significant improvement of the reasoning time compared to a naive approach.

1. Introduction

Ontologies [Staab and Studer, 2010] provide the vocabulary necessary to represent knowledge related to a specific domain by formally defining concepts, relationships between these concepts, and instances of these concepts. In recent years, ontologies have been widely used in many fields such as bioinformatics [Consortium, 2014] or recommender systems [Middleton et al., 2004, Kang et al., 2014], allowing experts to easily exchange knowledge and reason on well-structured data.

Similarly, ontology reasoning has become a well-adopted technique for the maintenance and query of ontologies, or the matching of different ontologies. Thanks for flexible and expressive rule languages [Horrocks et al., 2004] and powerful reasoners [Sirin et al., 2007, Shearer et al., 2008], rules can easily be written and maintained by non-experts and can easily be adapted to changing ontologies. The main downside of this expressiveness and flexibility is its reasoning complexity. Ontology rules, similarly to first-order logic, use variables that can be assigned to the entities of the ontology. Reasoning then consists in evaluating the rules by considering each combination of entities to the variables and checking if the conditions of the rules are true. This is a computationally complex operation which have been shown to limit the applicability of ontologies to real-time systems [Hashimoto et al., 2017].

In some applications, entities are constantly changing and reasoning must be performed as fast as possible, making typical reasoning methods based on first-order logic too slow. Instead, decision tables can be used to rapidly compute decisions but are hard to maintain. It is possible to transform rules used in ontology to decision table through a process known as propositionalization [Krogel et al., 2003]. This process has several downsides, requiring to already know the values for the variables (i.e., the entities of our ontology) and generating a decision table of exponential size.

In this paper, we are interested in allowing efficient reasoning with ontologies through the use of decision tables. We propose a method to transform rules to propositional form using placeholders instead of real entities, allowing us to perform a large part of the computation in advance. Placeholders are generated with classes corresponding to the variables used in the rules, creating a correspondance between placeholders and variables. During reasoning, we only need to map the real-world entities to the placeholders instead of mapping them to all the variables of the rules. In our experiments, we use a car ontology designed for automated-driving where fast reasoning time is critical to avoid accident. We show that our method is much faster than the naive propositionalization used when directly reasoning in first-order logic and that even complex situations can be handled within 122ms.

Contact: maxime-clement@nii.ac.jp

2. Preliminaries

In this section, we first briefly explain how we represent ontologies using first-order logic. We then describe the naive propositionalization necessary to use decision tables to reason with ontology rules.

2.1 Ontologies

An ontology formally defines the vocabulary necessary to describe some specific domain, usually in the form of a set of entities, classes, and properties. Entities are used to represent real-world objects, classes are collections of entities with a common type, and properties are relationships between entities.

We use first-order logic to formally write ontologies such that entities are constants, classes are unary predicates, and properties are binary predicates.

Example 2..1 (Ontology)**.** The knowledge "the car A is in front of the bike B" is written as

$$\textsc{Car}(a), \textsc{Bike}(b), \textsc{InFrontOf}(a, b)$$

where a and b are entities, Car and Bike are classes, and InFrontOf is a property.

In order to reason with an ontology, we use rules of the form $R = \mathit{body} \rightarrow \mathit{head}$ where *body* and *head* are conjunctions of literals. Similary to the logical implication, this means that if the content in *body* is true, then the content in *head* should also be true. In order to have general rules that can be applied to multiple entities, variables that can be replaced with any entity of the ontology are used. We refer to the set of variables used in a rule R as $var(R)$.

Example 2..2 (Rule)**.** The rule "if something is a car, it is a vehicle" is written as

$$\textsc{Car}(X) \rightarrow \textsc{Vehicle}(X)$$

where X is a variable.

2.2 Propositionalization

Propositionalization is the process of converting information from relational form (like in ontologies) to propositional form made exclusively of true or false propositions. Rules in propositional form corresponds directly to a decision table where each rule represents a row such that its body is the set of conditions and its head the set of actions.

To fit rules written in first-order logic into a decision table, each assignment of entities to the variables must be considered. Once an assignment is given, the rules can be converted to propositional form where each predicate is replaced by a proposition corresponding to the assignment. Similarly, knowledge about entities is also propositionalized, giving us a set of true propositions that can be used to determine whether the body of a rule is true or not.

Example 2..3 (Rule Propositionalization)**.** Given a rule $\textsc{Car}(X) \to \textsc{Vehicle}(X)$ and an assignment $\{X : a\}$ meaning entity a is assigned to variable X, we generate the propositional rule

$$\textsc{Car}_a \to \textsc{Vehicle}_a$$

Assuming the knowledge $\textsc{Car}(a), \textsc{Bike}(b)$, we generate the propositions

$$\textsc{Car}_a, \textsc{Bike}_b$$

We can now evaluate the previous rule and determine that its body is true since $\textsc{Car}_a$ is true, which means that $\textsc{Vehicle}_a$ is also true.

There exists several limitions with such *naive* propositionalization. First, the rules generated are for some specific entities, meaning that if entities change, then new rules should be generated. Second, each combination of entities to variables need to be considered, which quickly leads to an intractable number of propositional rules.

3. Typed Propositionalization

In order to improve the conversion from ontology rules to decision table, we propose *typed propositionalization* which uses a preprocessing step that performs propositionalization without the need for entities to be defined. The idea of this method is to instantiate the rules in advance, using placeholders instead of the real entities. When reasoning, real entities only need to be assigned to these placeholders, which requires less computation and makes the reasoning faster.

Definition 3..1 (Typed Placeholder)**.** We define $T = \{\mathrm{C}_0, \mathrm{C}_1, \ldots\}$ as a typed placeholder where C_i is an unary predicate (i.e., a class).

For each rule R, a placeholder is associated to each variable using the classes of the variable. For each variable $v \in \mathit{var}(\mathit{body})$, a placeholder $T_v = \{\mathrm{C} | \mathrm{C}(v) \in \mathit{body}\}$ is generated where *body* is the body of rule R. When all the variables in R have a non-empty placeholder, we say that R is *well-typed.*

The rule R is then converted to propositional form $R_{\mathcal{T}}$ using the assignment $\{v : T_v | v \in \mathit{var}(\mathit{body})\}$ such that each property $\mathrm{P}(x, y)$ is replaced by a proposition $\mathrm{P}_{T_x T_y}$ and each class predicate in the body is removed.

Example 3..1 (Typed Propositionalization)**.** Given a well-typed rule

$$\textsc{Car}(c) \land \textsc{Vehicle}(v) \land \textsc{InFrontOf}(v, c) \to \textsc{Warning}(c)$$

we generate the placeholders:

- $T_c = \{\textsc{Car}\}$ (placeholder for c);
- $T_v = \{\textsc{Vehicle}\}$ (placeholder for v).

We then generate the rule $R_{\mathcal{T}}$.

$$\textsc{InFrontOf}_{T_v T_c} \to \textsc{Warning}_{T_c}$$

After this preprocessing step, we have the set of all placeholders $\mathcal{T}$ and the generated propositional rules.

4. Reasoning with Typed Propositionalization

Once the rules have been propositionalized using placeholders, we are ready to perform reasoning with real entities. To reason using our typed-rules, we need to match the entities to their corresponding placeholder, generate the corresponding typed-facts, and execute the rules. We thus separate the reasoning into 4 phases.

1. Entity matching: match compatible entities with the placeholder.
2. Knowledge propositionalization: create propositional facts based on the matching.
3. Typed Reasoning: trigger the rules based on the created propositional facts.
4. Interpretation: transpose the new facts for their corresponding entities (instead of the placeholders).

Because several entities can correspond to a same placeholder, there might be several assignment to consider and steps 2-4 need to be repeated for each possible assignment.

4.1 Entity Matching

Entity matching consists in creating, for each placeholder T, the set of entities $E_T = \{e | T \subseteq \mathit{classes}(e)\}$ where $\mathit{classes}(e)$ is the set of classes of the entity e. If for at least one placeholder T there is $|E_T| > 1$, then we need to consider multiple assignments of entities to the placeholders. The number of total assignments to consider is $\prod_{T \in \mathcal{T}} |E_T|$. We refer to an assignment A as a set of pair $(T : e)$ meaning that entity e is mapped to placeholder T.

Example 4..1 (Entity Matching)**.** Given the placeholders $\mathcal{T} = \{T_c = \{\textsc{Car}\}, T_v = \{\textsc{Vehicle}\}\}$, and two entities a and b such that $\textsc{Car}(a), \textsc{Vehicle}(a), \textsc{Bike}(b), \textsc{Vehicle}(b)$, we generate the set of matching entities for each placeholder: $E_{T_c} = \{a\}$; $E_{T_v} = \{a, b\}$. We can thus consider two assignments $A_1 = \{T_c : a, T_v : a\}$ and $A_2 = \{T_c : a, T_v : b\}$.

Entities	Naive Propositionalization		Typed Propositionalization	
	Propositional Rules	Reasoning Time (ms)	Propositional Rules	Reasoning Time (ms)
6	105360	591	300	3
8	616560	6230	300	4
10	2436840	31419	300	4
12	OOM	-	300	7
14	OOM	-	300	13
16	OOM	-	300	35
18	OOM	-	300	70
20	OOM	-	300	122

Table 1: Comparison of the reasoning time and number of propositional rules using naive and typed propositionalization with 26 placeholders and 300 rules. OOM indicates settings where we ran out of memory.

4.2 Knowledge Propositionalization

Given an assignment A, we propositionalize all knowledge by replacing references to an entity e by its assigned placeholder T.

Example 4..2 (Knowledge Propositionalization)**.** Given the assignment $A_2 = \{T_c : a, T_v : b\}$, and the knowledge $\textsc{Car}(a), \textsc{Vehicle}(a), \textsc{Bike}(b), \textsc{Vehicle}(b), \textsc{InFrontOf}(b, a)$, we generate the propositional knowledge $\textsc{Car}_{T_c}, \textsc{Vehicle}_{T_c}, \textsc{Bike}_{T_v}, \textsc{Vehicle}_{T_v}, \textsc{InFrontOf}_{T_v T_c},$

4.3 Typed Reasoning

Now that we converted facts expressed for our entities to facts expressed for the placeholders, we can evaluate each propositional rule $R_{\mathcal{T}}$.

Example 4..3 (Reasoning)**.** Given the propositional rule from Example 3..1, and the propositional knowledge $\textsc{Car}_{T_c}, \textsc{Vehicle}_{T_c}, \textsc{Bike}_{T_v}, \textsc{Vehicle}_{T_v}, \textsc{InFrontOf}_{T_v T_c},$ we obtain the new knowledge $\textsc{Warning}_{T_c}$ since the body of the rule ($\textsc{InFrontOf}_{T_v T_c}$) is part of our knowledge base.

4.4 Interpreting the Results

Once all rules have been evaluated for an assignment A, we need to convert the resulting predicates which are expressed for the placeholders $\mathcal{T}$ into first-order predicates expressed for the real-world entities. This conversion is the reverse of the knowledge propositionalization where we now replace each placeholder T by its corresponding entity e in the assignment A.

Example 4..4 (Interpretation)**.** Given the new knowledge $\textsc{Warning}_{T_c}$ obtained with the assignment $A_2 = \{T_c : a, T_v : b\}$, we interpret it as $\textsc{Warning}(a)$.

Once this interpretation has been done for the results obtained with each possible assignment, we obtained all the possible implications for the current state of the ontology.

5. Experimental Results

We now present results obtained using the ADAS ontology[*1] [Zhao et al., 2017] written in OWL2 [Grau et al., 2008, Motik et al., 2009] and rules written in SWRL [Horrocks et al., 2004]. We compare the naive propositionalization presented in Section 2.2 and typed propositionalization presented in Section 3. and 4.. Both methods were implemented in Java using the OWLAPI [Horridge and Bechhofer, 2011] to interact with the ontology, using an Intel Core i7-7700K running at 4.20GHz and with 1GB of memory dedicated to the JVM. We use a set of 300 SWRL rules with the number of variables ranging from 2 to 6. This set was previously used for testing the speed of ontology reasoners [Hashimoto et al., 2017] and we manualy made minor modifications to ensure all rules are well-typed. This dataset generated 26 placeholders using our typed-propositionalization.

Table 1 shows the reasoning time (average over 10 runs) and number of propositional rules for the naive and typed propositionalization varying the number of entities. For naive propositionalization, we first observe the very high number of propositional rules generated, going from 105360 with 6 entities to more than 2 millions with 10 entities. Generating and evaluating such high number of rules also leads to a high reasoning time, taking 591ms with 6 entities and more than 30s with 10 entities. This leads the naive propositionalization to require more memory than available with more than 10 entities.

For typed propositionalization, we see that the number of propositional rules is always 300, thanks for our approach using placeholders which always produces one propositional rule per SWRL rule. Even if the number of propositional rules does not change, we observe that the reasoning time increases exponentially with the number of entities. This is due to the evaluation of the rules needing to be repeated for each possible assignment of entities to the placeholders. Despite the exponential increase of the reasoning time, our proposed approach allows to obtain runtimes that are appropriate for real-time reasoning, reaching only 122ms with 20 entities.

These results show that it is possible to convert rules for ontology reasoning into the format of decision tables. With typed propositionalization, little memory is required to store the rules in propositional form and the reasoning remains fast enough for real-time decision making in situations of medium complexity (up to 20 entities).

*1 http://ri-www.nii.ac.jp/ADAS/index.html

6. Conclusion and Future Works

Reasoning with ontologies is a complex problem which can limit its use with real-time applications. In this paper, we proposed a typed propositionalization technique to efficiently represent ontology rules as decision tables, allowing a faster reasoning better suited for real-time applications. Our experiments showed that our proposed approach leads to a much better reasoning time and memory complexity compared to a naive approach.

In future works, we will extend our technique to also define placeholders based on some relations, in addition to using the classes of the variables. Furthermore, we plan on studying ways to compress and decompose the rules once in propositional form. We believe methods based on Binary Decision Diagrams [Akers, 1978] or Zero-Suppressed Decision Diagrams [Minato, 1993] to be the most promising.

Acknowledgements

This work was partially supported by the New Energy and Industrial Technology Development Organization (NEDO).

References

[Akers, 1978] Akers, S. B. (1978). Binary decision diagrams. *IEEE Transactions on computers*, (6):509–516.

[Consortium, 2014] Consortium, G. O. (2014). Gene ontology consortium: going forward. *Nucleic acids research*, 43(D1):D1049–D1056.

[Grau et al., 2008] Grau, B. C., Horrocks, I., Motik, B., Parsia, B., Patel-Schneider, P., and Sattler, U. (2008). Owl 2: The next step for owl. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309–322.

[Hashimoto et al., 2017] Hashimoto, K., Ishida, Y., Ichise, R., Wagatsuma, H., and Tamukoh, H. (2017). A fundamental study on the performance of the logical reasoning system by using the semantic web techniques when it is applied to real vehicle automated driving to detect possible dangers predictively. In *Proceedings of the 61st Annual Conference of the Institute of Systems, Control and Information Engineers*, volume 324, pages 681–684. In Japanese.

[Horridge and Bechhofer, 2011] Horridge, M. and Bechhofer, S. (2011). The owl api: A java api for owl ontologies. *Semantic Web*, 2(1):11–21.

[Horrocks et al., 2004] Horrocks, I., Patel-Schneider, P. F., Boley, H., Tabet, S., Grosof, B., Dean, M., et al. (2004). Swrl: A semantic web rule language combining owl and ruleml. *W3C Member submission*, 21:79.

[Kang et al., 2014] Kang, Y.-B., Pan, J. Z., Krishnaswamy, S., Sawangphol, W., and Li, Y.-F. (2014). How long will it take? accurate prediction of ontology reasoning performance. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 80–86.

[Krogel et al., 2003] Krogel, M.-A., Rawles, S., Železný, F., Flach, P. A., Lavrač, N., and Wrobel, S. (2003). Comparative evaluation of approaches to propositionalization. In *Proceedings of the 13th International Conference on Inductive Logic Programming*, pages 197–214. Springer.

[Middleton et al., 2004] Middleton, S. E., Shadbolt, N. R., and De Roure, D. C. (2004). Ontological user profiling in recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):54–88.

[Minato, 1993] Minato, S.-i. (1993). Zero-suppressed bdds for set manipulation in combinatorial problems. In *Proceedings of the 30th international Design Automation Conference*, pages 272–277. ACM.

[Motik et al., 2009] Motik, B., Patel-Schneider, P. F., Parsia, B., Bock, C., Fokoue, A., Haase, P., Hoekstra, R., Horrocks, I., Ruttenberg, A., Sattler, U., et al. (2009). Owl 2 web ontology language: Structural specification and functional-style syntax. *W3C recommendation*, 27(65):159.

[Shearer et al., 2008] Shearer, R., Motik, B., and Horrocks, I. (2008). Hermit: A highly-efficient owl reasoner. In *OWLED*, volume 432, page 91.

[Sirin et al., 2007] Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical owl-dl reasoner. *Web Semantics: science, services and agents on the World Wide Web*, 5(2):51–53.

[Staab and Studer, 2010] Staab, S. and Studer, R. (2010). *Handbook on ontologies*. Springer Science & Business Media.

[Zhao et al., 2017] Zhao, L., Ichise, R., Liu, Z., Mita, S., and Sasaki, Y. (2017). Ontology-based driving decision making: A feasibility study at uncontrolled intersections. *IEICE Transactions on Information and Systems*, E100.D(7):1425–1439.