SOGrounder: Modelling and Solving Second-Order Logic (Extended Abstract)*

Matthias van der Hallen[1][0000−0003−1893−9369]** and Gerda Janssens[1][0000−0002−7836−1856]

KU Leuven: Department of Computer Science
Celestijnenlaan 200A, 3001 Heverlee (Leuven), Belgium
firstname.lastname@kuleuven.be

1 Introduction

A knowledge representation (KR) system allows *declarative* modelling of real-world knowledge, supporting one or more *inferences* to be performed on the model, e.g. satisfiability checking, model expansion or optimisation of problems within the knowledge domain. While many KR systems find their roots in logic, most of these systems restrict themselves to first-order logic over finite domains.

In the following sections, we present a KR system covering second-order logic, with an implementation based on the well-known, two phased *ground-and-solve* paradigm, making it possible to model more expressive constraints, e.g. co-NP constraints. Section 2 offers some insight into the implementation by detailing the most important part of the grounding transformation, while Section 3 provides an example which shows how to use the resulting KR system.

2 Grounding Second-Order Logic

As second-order logic, with a descriptive complexity of PH [2], is a more expressive language than first-order logic, we propose the more expressive *Quantified Boolean Formulas* (QBF) as grounding target, rather than the less expressive *SAT* or *ground (disjunctive) ASP*. Quantified Boolean Formulas are formulas of the form

$$\mathcal{Q}_1\overline{x}_1 \dots \mathcal{Q}_n\overline{x}_n.\phi$$

where $\mathcal{Q}_n$ are quantifiers $\forall$ or $\exists$, $\overline{x}_n$ are propositional variables and ϕ is a quantifier-free, propositional formula.

The main idea of grounding second-order logic formulas over finite domains to QBF is to replace all predicate atoms $p(\overline{x})$ by propositions $p_{\overline{x}}$ and replace all quantifications $\mathcal{Q}p$ by quantifications $\mathcal{Q}p_{\overline{x_0}}, \dots, \mathcal{Q}p_{\overline{x_n}}$ with $\overline{x_i}$ in the domain $\mathcal{D}_p$ of p. For example, $\forall p : \phi \wedge p(1)$ with domain $\mathcal{D}_p$=$\{1, 2\}$ would become $\forall p_1 : \forall p_2 : \phi[p(1)/p_1, p(2)/p_2] \wedge p_1$ [1].

* Copyright © 2019 for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

** Matthias van der Hallen is supported by a Ph.D. fellowship from the Research Foundation - Flanders (FWO - Vlaanderen).

Listing 1.1. SO model for *Critical Friendship*

```
1        type Person.                 % The Person type, corresponding to nodes
2        type Car.                    % The Car type, corresponding to colours
3        dislike :: (Person, Person). % A predicate encoding dislikes
4        c1      :: Person.           % A person constant
5        c2      :: Person.           % A second person constant
6        c1 ≠ c2.
7        ∃f ::(Person)→Car:  ∀ a,b ::Person: dislike(a,b)⇒f(a)≠f(b).
8        ∀f ::(Person)→Car: (∀ a,b ::Person: dislike(a,b)⇒f(a)≠f(b)) ⇒ f(c1)=f(c2).
```

3 Modelling with Second-Order Logic

Consider the *Critical Friendship* problem: n persons want to travel to the same location, and decide to carpool. Some of them, however, dislike each other and do not want to be in the same car. By encoding these dislikes in a graph, we can employ k-colouring, with k the number of cars, to show that a solution exists. Additionally, we are interested in *critical friendship* pairs: Two distinct persons for whom it holds that if they disliked each other, *no* k-colouring would exist.

Listing 1.1 provides a model for the *Critical Friendship* problem: Lines 1-5 specify the *vocabulary*, declaring two types (`Person`s and `Car`s), and three logic symbols: The `dislike` relation and the two `Person`s `c1`, `c2`.

Lines 6-8 form the *theory*: second-order formulas, called sentences, with no free variables except symbols declared in the vocabulary. Specifically, line 6 simply ensures that `c1` and `c2` are distinct `Person`s. The constraint of Line 7 ensures that a function `f` representing a k-colouring exists. After rewording the critical friendship constraint to "Any valid k-colouring maps `c1` and `c2` to the same car", it is clearly encoded by Line 8.
Note the high similarity between the NP constraint of k-colouring *existence* (Line 7) and the co-NP critical friendship constraint, which asserts something for *all* k-colourings (Line 8). It is this co-NP property for which the *Critical Friendship* problem *requires* second-order logic: first-order cannot ensure that a property holds *for all* functions, it cannot express universal function quantification.

Having constructed a *Knowledge Base* (KB) by specifying a vocabulary and a theory, we can provide instance-specific data and use inferences to solve queries; e.g. after adding the following instance-specific data, we can use model expansion to find that the friendship between `Bob` and `Charles` is critical:

```
Person  = {Alice; Bob; Charles; David}.
Car     = {Astra; Berlingo}.
dislike = {Alice, Bob; Bob, David; Charles, David}.
```

4 Results and Future Work

Our system is available online (`http://dtai.cs.kuleuven.be/krr/SOGrounder/`). In previous work [1], this system was tested with different QBF solvers on the *strategic companies* problem, a well-known problem in ASP and QBF communities. QBF solvers, combined with our grounder system performed competitively with *clasp*, a well-known ASP solver. In current work, we are extending the system with (1) new operators to accommodate counterfactual reasoning, and (2) count and sum aggregates to allow for more numeric operations.

References

1. van der Hallen, M., Janssens, G.: Sogrounder: Modelling and solving second-order logic. In: KR. pp. 72–77. AAAI Press (2018)
2. Immerman, N.: Descriptive complexity. Graduate texts in CS, Springer (1999)