First-Order Logic Concept Symmetry for Theory Formation

Pedro Torres* and Simon Colton
Computational Creativity Group
Department of Computing, Imperial College London, UK
ptorres,sgc@doc.ic.ac.uk

1 Introduction

SURICATA (Torres and Colton, 2008) is a hybrid automated theory formation system which uses both production rules and structured language biased search to produce new concepts and make conjectures about those concepts. The idea of implementing a highly-configurable theory formation system able to work with arbitrary first-order production rules started from the observation that it was helpful for users of automated theory formation systems to be able to define their own production rules. We implemented a generic first-order production rule (Torres and Colton, 2006) for the HR system (Colton, 2002) and showed how new user-defined production rules led to the discovery of novel conjectures in quasigroup theory.

During theory formation, SURICATA searches for regularities in newly formed concepts — defined as first-order logic (FOL) formulas — in order to make conjectures. Currently, these conjectures arise by comparing the available examples of concepts (see final part of section 2). To improve SURICATA's ability to make conjectures, we explore here a large set of new and relevant regularities which arise from intrinsic symmetries of the concepts under study. An example of concept symmetry is $\forall x, y, z.(\phi(x, y, z) \leftrightarrow \phi(y, x, z))$. The motivation for studying these regularities is two-fold: (i) the symmetry group of a set is traditionally a useful mathematical tool to capture the properties of a set and hence we hope that the characterisation of a concept through its symmetries will lead to improved ways to search for concepts; (ii) the symmetries studied here partially propagate through FOL formulas, and hence through FOL production rules, decreasing the time complexity of the search for symmetries in the space of concepts. Group theory has been previously applied to detecting structural symmetries of models of concurrent systems (Donaldson and Miller, 2005).

Given a set of concepts with some particular symmetries, new concepts built from them will present symmetries which we know to be true by construction, without having to actually prove them: they come for free. To the best of our knowledge, the exact way in which these symmetries propagate within first-order logic and which symmetries are in fact relevant for theory formation, has not been studied in detail. Our goal is to get a better understanding of how exactly symmetries propagate and how that can be exploited within theory formation. We present preliminary results here.

2 First-order Theory Formation Setting

A *concept* is a relation between any n types. We will consider here concepts of the form $\phi(x_1, \ldots, x_n)$, where ϕ is some first-order logic formula with free variables $x_1, \ldots, x_n$. We write $x_i : \tau_i$ to say that variable x_i *has type* τ_i and, if each x_i has type τ_i, we say that concept $\phi(x_1, \ldots, x_n)$ has type $\tau_1 \times \cdots \times \tau_n$. To make types explicit, we write $\phi(x_1 : \tau_1, \ldots, x_n : \tau_n)$. We use the symbol $\mathcal{C}$ for the set of all concepts and $\mathcal{C}[\tau]$ for the set of all concepts of type τ.

A *production rule* is a function $\pi : \mathcal{D} \rightarrow \mathcal{C}$, where $\mathcal{D} \subseteq \mathcal{C}^q$, for some q. Suppose we have $\phi(x : \tau_1, y : \tau_2)$ and $\varphi(y : \tau_2, z : \tau_3)$. Then, for instance, we can define a production rule as: $(\phi(x, y), \varphi(y, z)) \mapsto \exists y.(\phi(x, y) \wedge \varphi(y, z))$. In this expression, ϕ and φ act as placeholders for concepts of the type described.

From an initial set of background concepts $\{\phi_1, \ldots, \phi_k\}$, by recursively applying production rules with appropriate domains, we can define new concepts. Given a set of initial concepts $\mathcal{C}_0$ and a set of production rules Π, we recursively define $\Pi^*(\mathcal{C}_0)$ as the set of all concepts which: (i) are in $\mathcal{C}_0$ or (ii) can be written as $\pi(\phi_1, \ldots, \phi_n)$ where π is in Π and every ϕ_i is in $\Pi^*(\mathcal{C}_0)$. The set $\Pi^*(\mathcal{C}_0)$ can be described as the set of all concepts that can be produced from the concepts in $\mathcal{C}_0$ by using the production rules in Π. If we fix $\mathcal{C}_0$, the set Π defines exactly which concepts are in the search space.

One way of making conjectures in a production rule based system is to compare the concepts produced. If at some point during theory formation, we have two concepts of the same arity and types for which all the available examples of one are also examples of the other and vice versa, it is possible to conjecture that they are equivalent. Similar situations may lead to implication and non-existence conjectures. The conjectures that such a system may come up with are therefore all of the form $[\forall \vec{x}.(\phi_1(\vec{x}) \rightarrow \phi_2(\vec{x}))]$, $[\forall \vec{x}.(\phi_1(\vec{x}) \leftrightarrow \phi_2(\vec{x}))]$ or $[\forall \vec{x}.\neg\phi_1(\vec{x})]$ where $\vec{x}$ is some set of variables and ϕ_1 and ϕ_2 are concepts in $\Pi^*(\mathcal{C})$.

*First author supported by Fundação para a Ciência e a Tecnologia, grant SFRH/BD/12437/2003.

3 Concept Symmetries

Types of symmetries We will assume here that concept variables all have the same type, τ. If a concept has variables with different types, we can always consider the symmetries we are about to describe restricted to those variables of the same type. Symmetries in concepts can take several forms. We will consider the symmetries defined below.

1. Let $\mathcal{P}_n$ be the set of all permutations of the set $\{1, \ldots, n\}$ and let $\sigma \in \mathcal{P}_n$. Given an n-tuple of variables $\vec{x}$, we write $\sigma\vec{x}$ to denote the tuple of variables obtained by reordering $\vec{x}$ according to σ. A concept $\phi(\vec{x})$ of arity n has *variable symmetry* if there exists $\sigma \in \mathcal{P}_n$ such that $\forall\vec{x}.\,(\phi(\vec{x}) \leftrightarrow \phi(\sigma\vec{x}))$. E. g. $\forall x, y, z.(\phi(x, y, z) \leftrightarrow \phi(y, z, x))$.

2. A concept $\phi(\vec{x})$ has *variable collapse symmetry* if there exists a set of substitutions of each variable x_i by another variable x_j (i and j may be equal) such that $\forall\vec{x}.\,(\phi(S\vec{x}))$ is true, where S acts on $\vec{x}$ to perform the mentioned set of variable substitutions. E. g. $\forall x, y.(\phi(x, x, y))$ is true.

3. A concept $\phi(\vec{x})$ is said to have *ground symbol symmetry* if there exists a bijective function $f : \tau \to \tau$ such that $\forall\vec{x}.(\phi(\vec{x}) \leftrightarrow \phi(\hat{f}\vec{x}))$, where $\hat{f}$ is a function which acts as f on one of the components of $\vec{x}$ and leaves all other components unchanged E. g. $\forall x, y.(\phi(x, y) \leftrightarrow \phi(fx, y))$, with $\tau = \mathbb{Z}$ and $fx = -x$.

4. Let τ be a type and define a group $G = (\tau, *)$. We define the (left) g-action of G on itself as the function $\alpha_g : G \to G$ such that $\alpha_g(x) = g * x$. A concept $\phi(\vec{x})$ has *G-action symmetry* if $\forall g, \vec{x}.(\phi(\vec{x}) \leftrightarrow \phi(\hat{\alpha}_g\vec{x}))$, for some $\hat{\alpha}_g$, where $\hat{\alpha}_g$ is a function which acts as α_g on one of the components of $\vec{x}$ and leaves all other components unchanged. E. g. $\forall g, x, y.(\phi(x, y) \leftrightarrow \phi(x, g * y))$. Group action symmetries can be generalised to arbitrary algebraic structures.

For each symmetry on the examples, we can write a corresponding conjecture. Searching for symmetries corresponds to having additional sophisticated conjecture making techniques which do more than mere testing sets of examples for equality or inclusion. Each of these techniques actually searches for intricate patterns in the examples which would take various steps to find in the normal theory formation setting of "concept formation followed by elementary conjecture making techniques", if at all reachable.

Symmetry Propagation Suppose that concepts ϕ_1 and ϕ_2 satisfy, $\forall x, y.(\phi_1(x, y) \leftrightarrow \phi_1(y, x))$ and $\forall x, y.(\phi_2(x, y) \leftrightarrow \phi_2(y, x))$. Then, we know that any new concept formed from these two concepts will have some kind of symmetry (not necessarily the same). For example, concept ϕ defined as $\phi_1(x, y) \wedge \phi_2(x, y)$ would still preserve the same symmetry, i.e. $\phi(x, y) \leftrightarrow \phi(y, x)$, for all x and y. Moreover, more intricate concepts such as $\phi_1(x, y) \wedge \exists z.\,(\phi_3(z) \wedge \phi_2(y, x))$ where ϕ_3 is some arity 1 concept, will still preserve the same symmetry. In the general case, symmetry propagation is more involved as variables from different concepts can interact in non-trivial ways. Our main observation is that, since in an automated theory formation system all concepts are produced using logical production rules, if we know how symmetries propagate through FOL basic connectives, we will know symmetries of concepts in the theory formed without having to prove them. They will be true by construction. Knowing symmetries of initial concepts allows us to know that certain symmetries will be present in the final concepts. It does not, however, give us a full account of all the symmetries of the final concepts. To know the remaining symmetries of a concept we will still have to compute them. Nevertheless, if we take symmetry propagation into account, we need to check for fewer symmetries in total to get the complete picture.

Current work We are currently implementing a SURICATA module which computes all the symmetries of a concept and which, given some initial concepts and a set of production rules, computes the symmetries of all producible concepts making use of symmetry propagation knowledge. We hope to show that the use of this knowledge makes the computation of symmetries more time efficient. Using this new symmetry module we will be able to perform theory formation using a greedy search on the number of symmetries of concepts and, also, to search for concepts which present particular bespoke symmetries, which is a novel paradigm in theory formation.

References

S. Colton. *Automated Theory Formation in Pure Mathematics*. Springer-Verlag, 2002.

A F Donaldson and A Miller. Automatic symmetry detection for model checking using computational group theory. *Lecture Notes in Computer Science*, 3582:481–496, 2005.

P Torres and S Colton. Using Model Generation in Automated Concept Formation. In *Proceedings of ARW'06*, 2006.

P Torres and S Colton. Automated Meta-Theory Induction in Pure Mathematics. In *Proceedings of ARW'08*, 2008.