Towards Meta-Level Theory Formation

Pedro Torres* and Simon Colton

Department of Computing, Imperial College London, UK

180 Queen's Gate, Imperial College London, SW7 2AZ.

`ptorres,sgc@doc.ic.ac.uk`

Introduction

HR [1] is an Automated Theory Formation (ATF) system which has been described as a Descriptive Inductive Logic Programming system [4]. Given a set of initial concepts, each provided with a definition and a set of examples, HR forms a theory about them. HR uses a set of production rules to form new concepts and forms conjectures according to the examples it finds for new concepts.

We address here the question of whether a theory formation system can gain from analysing interesting established theories, before it attempts actual theory formation. The approach taken here is to improve the theory formation abilities of a theory formation system by allowing a prior learning stage where the system is allowed to consult existing well-established theories — so called *inspiring theories*. The aim of this project is therefore to test the hypothesis that theory formation can be improved by introducing a prior learning stage where inspiring theories can be consulted. We call this learning stage the *meta-learning stage* since the system is auto-configuring itself to perform a different learning task.

In a study involving algebraic structures [2], we extended our theory formation approach with a generic first-order production rule. By writing explicit definitions of first-order production rules, the user could add to the set of production rules used by HR. We extend this approach here, and we investigate how a search for sets of production rules could be carried out. If such a search can be efficiently implemented, this opens up the possibility of automatically generating production rule based theory formation systems tailored to a specific domain. Such tailoring will be enabled by the user presenting core background theory and interesting concepts derived from them, with our system able to generate production rule sets that produce the interesting concepts from the core ones, and hopefully additional interesting material.

First-order representation of production rules

Let $\{\tau_i\}_{i=1}^{N}$ be a collection of sets, which we will call *types*. We define a *concept* to be a relation between any n types. We call n the *arity* of the concept. We will consider here concepts of the form $\{(x_1, \ldots, x_n) \in \tau_1 \times \cdots \times \tau_n \mid \varphi\}$, where φ is some first-order logic formula for which the only free variables are $x_1, \ldots, x_n$ and any other variables are bounded. The above concept will be written as $[x_1, \ldots, x_n] : \varphi(x_1, \ldots, x_n)$. We write $\mathcal{C}$ for the set of all concepts.

A *production rule* is a function $\pi : \mathcal{D} \to \mathcal{C}$, where $\mathcal{D} \subset \mathcal{C}^q$, for some q, called the *arity* of π. A production rule can be defined by specifying the transformation it operates on the logical formulas defining the input concepts. Suppose we have $x \in \tau_1$, $y \in \tau_2$ and $z \in \tau_3$. We can define, for example, π as a function carrying out the mapping $([x, y] : \varphi(x, y), [y, z] : \varphi(y, z)) \mapsto [x, z] : \exists_y.(\varphi(x, y) \wedge \varphi(y, z))$.

From an initial set of background concepts $\{C_1, \ldots, C_k\}$, by recursively applying production rules with appropriate domains, we can define new concepts, e.g. $C = \pi_6(\pi_4(\pi_1(C_1, C_2), \pi_2(C_2, C_3)), \pi_5(\pi_3(C_4)))$. Given a set of initial concepts $\mathcal{C}$ and a set of production rules Π, we recursively define $\Pi^*(\mathcal{C})$ as the set of all concepts which: (i) are in $\mathcal{C}$ or (ii) can be written as $\pi(C_1, \ldots, C_n)$ where π is in Π and every C_i is in $\Pi^*(\mathcal{C})$. The set $\Pi^*(\mathcal{C})$ can be intuitively described as the set of all concepts that can be produced from the concepts in $\mathcal{C}$ by using the production rules in Π. If we fix $\mathcal{C}$, the set Π defines exactly which concepts are in the search space. The essential difference between language biases and sets of production rules is that the former usually let the user directly restrict the syntactical form of all the concepts formed, whereas the latter restrict the stepwise syntactical construction of new concepts from old ones.

One way of making conjectures in a production rule based system is to compare the concepts produced. Suppose we have two concepts of the same arity and types for which all the available examples which satisfy one also satisfy the other and vice versa. In these circumstances, it is possible to conjecture that they are equivalent. Similar situations may lead to implication and non-existence conjectures. The association rules that such a system may come up with are therefore all of the form $\left[\forall_{\overrightarrow{x}}.C_1(\overrightarrow{x}) \to C_2(\overrightarrow{x})\right]$, $\left[\forall_{\overrightarrow{x}}.C_1(\overrightarrow{x}) \leftrightarrow C_2(\overrightarrow{x})\right]$ or $\left[\forall_{\overrightarrow{x}}.\neg C_1(\overrightarrow{x})\right]$ where $\overrightarrow{x}$ is some set of variables and C_1 and C_2 are concepts in $\Pi^*(\mathcal{C})$.

It is possible to define a conjecture as a 0-arity concept. In this light, conjecture making techniques are simply particular production rules of arity 2 (they take two concepts) which yield concepts of arity 0 (i.e. conjectures). By using these

definitions, concepts and conjectures can be treated at the same level. Furthermore, we can define an order relation on the space of sets of production rules. Given a set of initial concepts, $\mathcal{C}$, and two sets of production rules, Π_1 and Π_2, we say that $(\Pi_1 \leq_{\mathcal{C}} \Pi_2)$ iff $\Pi_1^*(\mathcal{C}) \subseteq \Pi_2^*(\mathcal{C})$. In this case, Π_2 is said to be *richer* than Π_1.

Searching for production rules

The main idea we explore is that, instead of searching for concepts with a fixed set of production rules, we can start by analysing examples of previously formed (interesting) theories in a chosen domain and build a set of production rules tailored to that particular domain.

In order to perform the search for sets of production rules, we have defined three high-level algorithms: (i) the EX-PANDER algorithm for computing $\Pi^*(\mathcal{C})$, up to some defined depth, from user-defined $\mathcal{C}$ and Π; (ii) the PARROT algorithm which takes a large set of formed concepts, $\mathcal{C}$, all ultimately constructed from some elementary concepts $\mathcal{C}_0$, and defines a set of production rules, Π_0, such that $\Pi_0^*(\mathcal{C}_0) \supseteq \mathcal{C}$; (iii) the ENRICHER algorithm, which takes a set of production rules and iteratively enriches this set.

EXPANDER is a simple algorithm which considers all legal sequential applications of production rules up to some prescribed depth. In the PARROT algorithm, each concept, C, in the user-given set of concepts, $\mathcal{C}$, can be expressed as a first order formula making use of some old concepts $\mathcal{C}_0(C)$. It is therefore possible to craft a production rule which takes the concepts $\mathcal{C}_0(C)$ and produces C. If this is done for all the concepts in $\mathcal{C}$, we get a set of production rules, Π_0, which can generate all of the concepts in $\mathcal{C}$ from the elementary concepts they are based on. Since we have put concepts and conjectures on the same level, the above reasoning means that, if the user includes conjectures (or, possibly, theorems) in the set of concepts $\mathcal{C}$, the production rule set given by PARROT will also be able to make those conjectures. If the set of concepts, $\mathcal{C}$, is very restricted, the production rule set output by PARROT may not be able to generate much more than $\mathcal{C}$ itself. If this is the case, we will want to enrich that set.

A simple way to enrich a set of production rules, Π, is to remove a production rule π and add two separate production rules π_1 and π_2 such that, for every C in the domain of π, we have $\pi(C) = \pi_1(\pi_2(C))$. In this example we have considered rules of arity 1 but this *decomposition* idea can be generalised to higher arities. Given a set of production rules, there is a considerable number of possible decompositions enriching that set. To facilitate the analysis of these decompositions we have defined a set of so called *primitive* production rules which are closely based on the syntax of first order logic. By using the primitive production rules, we can write different composite production rules using a common vocabulary and search for matching subparts.

The ENRICHER algorithm is given a set of production rules and performs a best-first search through all decompositions. It uses EXPANDER to compute the concepts which each particular decomposition adds and calculates a score for each decomposition based on measures defined by the user. For example, using this algorithm, the user can ask for an enrichment which favours concepts with higher applicability (i.e. higher proportion of which satisfy the concept).

In the presentation poster, we will give details of the three theoretical algorithms mentioned above and toy examples of their application. This will include the complete definition of the primitive production rules, which play an important role in the ENRICHER algorithm, and the analysis of some of their properties.

References

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

[2] P. Torres and S. Colton. Using model generation in automated concept formation. In *Proceedings of the Automated Reasoning Workshop*, 2006.

[3] P. Torres and S. Colton. Towards meta-level descriptive ILP. In *Proceedings of the 16th International Conference on Inductive Logic Programming*, 2006.

[4] S. Colton and S. Muggleton. Mathematical applications of Inductive Logic Programming. *Machine Learning (online first)* 2005.