Proving Producibility of Concepts

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

Overview

Ground-level theory formation Inductive Logic Programming (ILP) (Muggleton, 1991) addresses the problem of finding logical theories which explain given observations. In the non-monotonic setting, also known as descriptive ILP, systems search for classification rules (concepts) and association rules (conjectures) which are true of the given examples.

One approach to this problem, which underlies the HR system (Colton, 2002), considers *production rules* – ways of producing new concepts from old ones – as operators to perform search in the space of concepts and specific *heuristics* to decide which production rule to apply to which concepts.

Meta-level paradigm Currently, we are working on the more general problem of learning how to form theories from examples (Torres and Colton, 2006b). This consists in analysing a set of formed theories, taken as examples, and automatically inducing a complete descriptive ILP system from that analysis. In other words, our goal is to have a meta-system which, given such a set of formed theories, outputs an ILP system able to re-create those theories. We hope that in this learning process, the generated ground system will also be able to produce additional interesting concepts.

Two unsolved problems Two important questions, both relevant for the construction of the mentioned meta-system, are: (i) given a theory formation system and a concept, how can we establish a formal proof that such concept is in theory producible by the system? (ii) how can we induce production rules from produced concepts which are able to re-create those concepts? The first question is important in the context of generalising and specialising theory formation systems. Given that a system is unable to generate a specific concept, we may be interested in reformulating it so that it can. Moreover, it can be useful in verifying soundness of the meta-system: we can have a formal proof that the ILP system created by the meta-system satisfies the basic requirement of producing, at least, the concepts given as initial examples for learning. The second question is crucial for the construction of the meta-system itself. Clearly, a meta-system will be required to generate production rules from produced concepts.

Our contribution In a previous study (Torres and Colton, 2006a), we presented a general framework for using model generation to implement generic first-order production rules. We have taken this idea one step further by designing a general automated theory formation system which is able to deal with arbitrary sets of first-order production rules. In the same study, we also sketched a method of backwards proof to prove concept producibility which we will here put to practice. In this workshop we will address both unsolved problems mentioned above by (i) presenting an algorithm for proving producibility of concepts by our theory formation system and (ii) putting forward ideas on how this algorithm may be useful to induce production rules from examples of produced concepts.

Concept Producibility

General framework for concept formation A *concept* is defined as a first-order logic formula. We write $\mathcal{C}$ for the set of all concepts. The *arity of a concept* is its the number of free variables.

A *production rule* is a function $\pi : \mathcal{D} \rightarrow \mathcal{C}$, where $\mathcal{D} \subset \mathcal{C}^q$, for some q, called the *arity* of π. The set of all production rules is denoted by Π. A production rule is effectively a "higher-order" function which takes logical formulas as variables. It can also be described by a first-order formula. For example, if we write the formula $\phi(x,y) \wedge \psi(y,x) \rightarrow \phi(x,x)$, we may interpret it as the production rule which takes formulas (ϕ, ψ), both of arity 2, and outputs the new concept of arity 2 described by the formula $\phi(x,y) \wedge \psi(y,x) \rightarrow \phi(x,x)$.

A *theory formation system* is a function $\Theta : 2^{\mathcal{C}} \times 2^{\Pi} \rightarrow 2^{\mathcal{C}}$ (where 2^X means the set of all subsets of X) which takes a set of concepts $\mathcal{C}_0$ and a set of production rules Π_0 and returns the set of all concepts which can be produced from $\mathcal{C}_0$ by application of some combination of production rules from Π_0. In practice, $\Theta(C_0, \Pi_0)$ is restricted to one of its finite

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

subsets and a set of heuristics is used to enumerate that subset in a particular order, e.g. giving priority to concepts with specific characteristics such as being used in more conjectures or being involved in proofs which take a long time.

Concept production proof We are interested in deciding whether a certain concept can be produced by a theory formation system. Suppose we have a set of initial concepts $\mathcal{C}_0$ and a set of target concepts Δ. Suppose additionally that a set of production rules Π_0 is defined. We are interested in implementing a program that takes $(\mathcal{C}_0, \Delta, \Pi_0)$ and outputs deductions of all target concepts Δ from the initial concepts $\mathcal{C}_0$. A deduction of a target concept T is a particular composition of production rules that yields T starting only from concepts in $\mathcal{C}_0$. These deductions can be seen as *concept proofs* stating that the target concepts are reachable by the production rule set and providing an explicit production path for them.

HAL HAL is an interactive theorem prover based on the sequent calculus. It is introduced in chapter 10 of (Paulson, 1996) as an elementary introduction to theorem proving in the Edinburgh LCF tradition. HAL features a tactic for each inference rule in first-order logic. Given a goal to prove, we can interactively apply tactics individually to break a goal into subgoals, until we get to a state where no subgoals are left. There are also higher-level tactics which search for combinations of simpler tactics providing moderate automation.

The key observation for producibility Our problem of concept producibility is very similar to the one HAL addresses, but $\phi \vdash \psi$ now means "ψ can be produced from ϕ" instead of "ψ can be deduced from ϕ". The *semantics* is different. In this new setting, for each production rule in our initial set Π, we need to introduce the inference rule

$$\frac{\Gamma \vdash \Delta, \phi_1 \quad \cdots \quad \Gamma \vdash \Delta, \phi_n}{\Gamma \vdash \Delta, \pi(\phi_1, \ldots, \phi_n)}$$

and create a usable tactic from it. From this observation, we have created a new generic tactic in HAL, which takes an arbitrary production rule and generates the tactic described above, and removed all existing first-order logic tactics. We also created the remaining pieces to fit this into HAL framework. This modified version of HAL can now take a set of production rules and become a concept prover for that particular set. But there is still a problem to tackle regarding unification.

Higher-order unification problem HAL uses first-order unification. In our case, we will want to unify the output of a production rule, $\pi(\phi_1, \ldots, \phi_n)$ with an arbitrary first-order formula. This is more intricate as predicates may need to be unified as well, requiring a sort of higher-order unification. For example, consider the production rule

$$\pi : (\phi_1(x, y), \phi_2(z)) \mapsto (\phi_1(x, x) \wedge \phi_2(y)) \vee \phi_1(y, x) \tag{1}$$

and suppose we wish to prove producibility of the concept

$$((\psi_1(x) \wedge \psi_2(x)) \wedge \psi_3(y, y)) \vee (\psi_1(x) \wedge \psi_2(y)). \tag{2}$$

The unification algorithm will have to find that $\psi_1(x) \wedge \psi_2(x)$ needs to be unified with $\phi_1(x, x)$. We are concluding the implementation of an algorithm which performs this kind of unification for arbitrary production rules and concepts and will present our findings at the workshop.

Application to induction of production rules Suppose we start with the concept given by (2) and we want to generate a production rule which is able to recreate it. A possible answer would be precisely the rule given by (1). Again, this induction algorithm will have to rewrite $\psi_1(x) \wedge \psi_2(x)$ as some $\phi_1(x, x)$ and solving the unification problem described above will also be useful in this context.

References

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

S. Muggleton. Inductive logic programming. *New Generation Computing*, 8(4):295–318, 1991.

L. C. Paulson. *ML for the working programmer*. Cambridge University Press, 2nd edition, 1996.

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

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