Using Model Generation in Automated Concept 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 (Colton, 2002) is an Automated Theory Formation (ATF) system which makes extensive use of automated reasoning tools, in particular, the Otter theorem prover (McCune, 1990) and MACE model generator (McCune, 1994). 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. For instance, if the set of examples for the new concept is contained in the set of examples of some old concept, an implication conjecture is generated. Similarly, equivalence and non-existence conjectures can be made. We present here a general first order setting for concept production in ATF, explain a recently implemented user-customisable generic production rule, and report on results in pure mathematics obtained using the new rule. The new concept production method make novel use of MACE, thus extending the combination of reasoning techniques employed in theory formation.

A general framework for concept production

General setting within first order logic Let $\{A_i\}_{i=1}^n$ be a collection of sets, which we will call *types*. Define $\mathcal{T}$ to be the set of all possible cartesian products made up of types, e.g. $A_1 \times A_2$, $A_2 \times A_2 \times A_5$, etc. We define a *concept* to be a subset of an element of $\mathcal{T}$, which means, equivalently, that a *concept* is an m-relation defined between any m types. We call m the *arity* of the concept. We write $\mathcal{C}$ for the set of all concepts. We will here consider concepts of the form $\{(a_1, \ldots, a_n) \in A_1 \times \cdots \times A_n | \ \phi(a_1, \ldots, a_n)\}$, where ϕ is some first-order logical formula for which the only free variables are $a_1, \ldots, a_n$ and any other variables are bounded. We will omit types whenever they are understood from context and write the above concept as $[a, b] : \phi(a_1, \ldots, a_n)$.

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. The set of all production rules is denoted by Π. A production rule can be defined by choosing the transformation it operates on the logical formulas defining the old concepts. Suppose we have types A, B and C. We can define, for example, $\pi : (A\times B)\times(B\times C) \to A\times C$ as a function carrying out the mapping $([a, b] : \phi(a, b), [b, c] : \varphi(b, c)) \mapsto [a, c] : \exists_b.(\phi(a, b) \wedge \varphi(b, c))$. If we have examples for the old concepts, we can compute the examples of the new concept by applying the corresponding logical transformation. Starting with an initial pool of concepts $\{C_1, \ldots, C_k\}$, by recurringly applying production rules with appropriate domains, we can define new concepts, e.g. $C = \pi_3(\pi_2(C_1), \pi_1(C_2, C_3, C_4))$. The current version of HR has a set of 16 production rules, excluding the new rule introduced in this paper.

First order generic production rule In HR, all existing rules are fairly specific in the sense that, given a concept defined by some logical formula, ϕ, the logical transformation they operate always involves ϕ explicitly. In order to allow greater customisation by the user[1], the *first order generic production rule* was defined. This rule is a first step in defining, for algebraic domains, the general framework described above. It takes one concept of the form $[a_1, \ldots, a_n] : p(a_2, \ldots, a_n)$ and produces a new concept $[a_1] : q(a_2, \ldots, a_n)$ where the formula q can involve the formula p and is defined by the user. As an example, this rule can take the old concept $[a, b] : p(b)$ and produce the new concept $[a] : \exists_b.p(b)$. A more complex example could be to use the same old concept to produce the new concept $[a] : \forall_{b_1,b_2}.[p(b_1) \wedge p(b_2) \to (b_1 = b_2)]$. We can see the new production rule as a set of production rules, one for each first-order formula we choose. We will call each of these formulas a *parameterisation*. Before starting the theory formation process, a set of particular parameterisations needs to be defined by the user.

As can be seen from the examples above, parameterisations can take very distinct forms and we need a general way to compute the examples of the new concept produced by any parameterisation. To solve this, we coded HR to use MACE to compute the examples of the new concept. Suppose we start with some initial concept $[G, a_1, \ldots, a_n] : p(a_1, \ldots, a_n)$,

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

[1]One conclusion from the study reported in (Colton and Huczynska, 2003) is that users of HR would appreciate the ability to define production rules.

where G is a groupoid and $a_1, \dots a_n$ are elements in G, and we wish to compute $[G] : q(a_1, \dots, a_n)$. For each example of the initial concept, a particular groupoid G, we check if property q takes place. To do this, we supply MACE with: (i) the multiplication table of G, (ii) the definition of p and (iii) the definition of q involving p as per the parameterisation. We then use MACE in a model checking fashion to determine whether q holds for each G. The new concept produced will be formed by all groupoids G for which q holds.

Application The new production rule was used to explore algebraic structures, G, characterised by the single axiom $\forall_{a,b,c \in G}.(a * b) * c = b * (c * a)$. This is a special class of non-associative groupoids which we will call *star groupoids*. In order to study star groupoids, we defined two parameterisations for the new rule corresponding to the following logical transformations:

$$\begin{aligned} [G, b] : p(b) &\quad \mapsto \quad [G] : \exists_b . p(b) \\ [G, b] : p(b) &\quad \mapsto \quad [G] : \forall_{a,b,c} . p(a) \wedge p(b) \wedge (c = a * b) \rightarrow p(c). \end{aligned}$$

Using HR's standard production rules in conjunction with the first order generic rule parameterised as above, we found some non-obvious results. Examples of star groupoids of sizes 1 to 7 were initially generated using MACE. The conjectures obtained were subsequently proved by Otter as part of the automated theory formation process. After a number of theory formation steps we were able to extract the following theorems, all resulting from the introduction of the new production rule:

$$\begin{aligned} (i) &\qquad \forall_b . (\exists_c (c * b = b) \leftrightarrow \exists_d (b * d = b)) \\ (ii) &\qquad \forall_b . (\exists_c (c * b = c) \leftrightarrow \exists_d (b * d = d)) \\ (iii) &\qquad \forall_{a,b,c} . (a * a = a \wedge b * b = b \wedge c = a * b \rightarrow c * c = c) \\ (iv) &\qquad \forall_{a,b,c} . (\phi(a) \wedge \phi(b) \wedge c = a * b \rightarrow \phi(c)) \end{aligned}$$

where $\phi(a) \leftrightarrow \exists_c . (c * a = a)$ which means "a has a left identiy". These theorems can be respectively interpreted as: (i) every element of a star groupoid has a left identity iff it has a right identity; (ii) every element of a star groupoid is a right identity of some element iff it is a left identity for some (possibly different) element; (iii) the product of two idempotent elements is idempotent; (iv) the product of two elements which have a left identity, also has a left identity.

Concept production proof We are also interested in deciding whether a certain concept can be produced by a system of production rules. Suppose we have a set of initial concepts Γ and a set of target concepts Δ. Suppose additionally that a set of production rules Π is defined. In the spirit of theorem proving, we would like to implement a program that would take (Γ, Δ, Π) and output deductions of all target concepts Δ from the initial concepts Γ. A deduction of a target concept T is a particular composition of production rules that yields T starting only from concepts in Γ. 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.

The general idea we considered to construct such a program is to view the production rules in the backwards direction, from a goal to subgoals. The goal will be the new concept a production rule generates and the subgoals are the old concepts needed for that production (production rules of arity 1 will only have one such concept and therefore only one subgoal). The initial goals will be the concepts we wish to produce and we would like to reach a stage where every subgoal is a concept in Γ. At each stage, the subgoals are refined by further backward application of the production rules until all the remaining subgoals are initial concepts. Using this *refinement* process, the proof tree is constructed from the root upwards. One of the key elements for solving the problems caused by the existence of variables and quantification is the process of determining the appropriate substitutions which will require *unification*. We believe that the problem of automating such a process may raise complexity issues. However, the typing information production rules carry with them may be able to reduce complexity considerably.

References

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

S. Colton and S. Huczynska. The homer system. In *CADE-19*, 2003.

W. McCune. The Otter user's guide. Technical Report ANL/90/9, Argonne National Laboratories, 1990.

W. McCune. A David-Putnam program and its applications to finite first-order model search. Technical Report ANL/MCS-TM-194, Argonne National Laboratories, 1994.