Learning University Mathematics

Edmund Furse
Department of Computer Studies
University of Glamorgan
Pontypridd, Mid Glamorgan
CF37 1DL
UK
efurse@uk.ac.glamorgan

Abstract

This video demonstrates the program MU, the Mathematics Understander, which learns University level Pure Mathematics. The video is both concerned with MU's performance in learning and performing mathematics, and its underlying cognitive architecture, the Contextual Memory System, (CMS). The changes in knowledge representation during proof checking and problem solving are demonstrated graphically.

1. Introduction

Learning University level Mathematics is a complex task for both people and machines and difficult to model for three main reasons:

1. Students can learn any branch of mathematics, and mathematics is continually developing. Any learning model therefore has to be open in its representation.
2. Pure Mathematics is a very large domain over 2000 years old, and this means that a large amount of knowledge has to be acquired.
3. Understanding Mathematics Proofs and solving problems requires the ability to be able to retrieve the appropriate result at each step, and this expertise only comes with experience.

The Mathematics Understander (MU), [Furse, 1994] is a computational model of how students learn Pure Mathematics from texts written in a notation known as the Formal Expression Language (FEL), [Furse, 1990]. FEL is a very expressive language and capable of representing almost all branches of pure mathematics. MU has read texts in both Classical Analysis and Group Theory. An extract from an FEL text is shown in Figure 1, where a definition, theorem and proof are shown in Classical Analysis.

MU has no built in knowledge of mathematical results, and acquires all its knowledge from the reading of texts. This knowledge is of two sorts:

1. The mathematics results: definitions of concepts, theorems and lemmas.
2. Features which index the mathematical results, and enable easy retrieval of the appropriate result.

The features provide an open form of representation thus enabling MU to learn in an open domain [see Furse, 1993].

Definition 2.4 of converges
$«s_n»$ converges to α iff $\forall \varepsilon > 0\ \exists m\ \forall n > m\ |s_n - \alpha| < \varepsilon$

Lemma 2.1.1
not ($«(-1)^n»$ converges to 1)
Proof
1 RTP not ($«(-1)^n»$ converges to α)
2 Suppose to the contrary $«(-1)^n»$ converges to 1
3 $\Rightarrow \forall \varepsilon > 0\ \exists m\ \forall n > m\ |(-1)^n - 1| < \varepsilon$
by definition of converges
4 $\Rightarrow \exists m\ \forall n > m\ |(-1)^n - 1| < 1$ by substituting $\varepsilon = 1$
5 let $n = 2m + 1$
6 $\Rightarrow$ n is odd
7 $\Rightarrow (-1)^n = -1$
8 $\Rightarrow |(-1)^n - 1| = |-2|$ since $a = b \Rightarrow |a-1| = |b-1|$
9 = 2
10 $\Rightarrow 2 < 1$ by combining steps 4 and 9
11 $\Rightarrow$ contradiction
12 QED

Figure 1. Input to MU in FEL.

2. The Mathematics Task

The mathematics task consists of checking proofs and solving problems using the knowledge of mathematical results that has been already been acquired. Clearly there is more to understanding a mathematics text than this but this is the scope of MU's understanding,

A knowledge of the mathematical results is not sufficient either to check proofs or solve problems since there are many results that could be applied at any one step. It is also necessary to have some control knowledge with to choose the appropriate result. In theorem proving systems, such as Ammon [1992], this is often done by heuristics, but in MU the choice of result is largely driven by perceptual features.

MU performs proof checking just by use of its features to retrieve the appropriate result. As explained in the next section through the experience of checking proofs and solving problems the features change in their representation, so that gradually more specific features emerge which enable important results to be easily retrieved, thus overcoming the combinatorial choice. This also ensures that MU goes faster rather than slower with more knowledge, as the specialised features ensure that retrieval is faster.

In problem solving, MU uses four general purpose heuristics to give overall control: 1. Break a problem up into parts; 2. Suppose the left hand side; 3. Expand definitions; 4. Simplify expressions. A verbatim solution of MU using these heuristics to solve a problem in Group Theory is shown in Figure 2. The simplification process uses the CMS to control which results are applied. The above heuristics are not sufficient to solve all problems in mathematics, and research has shown that there is a need for specialised heuristics as well as the features used by MU. Morgan et al [1995] show how the heuristics can be learned from the analysis of worked examples.

3. Learning and the CMS

MU's learning is all performed by the Contextual Memory System (CMS) which controls all the storage and retrieval of the mathematical results in terms of dynamic features. Because of the open learning requirement, there are no built in features. Rather the CMS uses built in feature creation mechanisms to generate a large number of features from the input.

The CMS performs its learning in two stages:

1. Initial encoding of the mathematical results when MU reads the FEL statement in terms of features.
2. Continual updating of the features through the experience of proof checking and problem solving.

Feature creation works on a parsed form of the input. For example, the definition of abelian in Fig. 2 is parsed as

```
(<=> (abelian (cap g)) (and (group (cap g))
    (forall (variables a b)
  (propn (=> (and (member a (cap g)) (member b (cap g)))
                (=(*ab)(*ba)))))))
```

If this is treated as a tree, features can be constructed as subtrees. Formally, features are defined by:

```
<feature> ::= <position><specifierxtypexterm>
<position> ::= LHS- I RHS- I null
<position> ::= LHS-<position> I RHS-<position>
<specifier> ::= IS- I HAS-
<type> ::= FORM- I TERM-
```

Since the <term> is as open as the input, this results in an infinite feature space. In this example there are many features including general features such as: has-form-[=_a_bJ, is-form-[<=*_a_b], has-form-[member_a_b] and more specialised features such as: lhs-has-form-[abelian_[cap_a]], rhs-has-form-[and_[group_a]_b] is-form-[»_[abelian_aL[and_b_c]] and only about 10 features are used for the first encoding.

The initial encoding into features of a result may not be very efficient as at the time it is impossible to know which features will turn out to be salient. With subsequent experience, proof checking and problem solving will provide probes with which to do retrieval from the CMS. Often such retrievals will recall a result which cannot be applied to the current step, as well as the correct result, and these are termed failures. The CMS learns from this experience in three ways:

1. Features which were used in the searching using the probe but are not currently linked to the found item are termed 'uncomputed features'. These features are now linked to the item.
2. Features which index the found item but not the failures are termed 'useful features', and have their energies increased as well as the strength of the link.
3. It is also possible to create new features, unknown at the time to the CMS, which distinguish the found item from the failures.

Definition of abelian
G is abelian iff G isa group and $\forall a,b \in G$ ab = ba

Problem 2.3.3
Prove (G isa group and $\forall a, b$ a $\in$ G
and b $\in$ G $\Rightarrow$ $(ab)^2 = a^2b^2$) $\Rightarrow$ G is abelian
Suppose G isa group
and $\forall a, b$ a $\in$ G and b $\in$ G $\Rightarrow$ $(ab)^2 = a^2b^2$
RTP G is abelian
RTP G isa group and $\forall a, b$ a $\in$ G and b $\in$ G ab = ba
by definition of abelian
Part 1
RTP G isa group
Follows logically
QED Part 1
Part 2
Suppose a $\in$ G and b $\in$ G
RTP ab = ba
Now $(ab)^2 = a^2b^2$
$\Rightarrow$ (ab)(ab) = (aa)(bb) since $x^2 = xx$
$\Rightarrow$ a((ba)b) = a((ab)b) since (ab)(cd) = a((bc)d)
$\Rightarrow$ (ba)b = (ab)b since ab = ac $\Rightarrow$ b = c
$\Rightarrow$ ba = ab since ba = ca $\Rightarrow$ b = c
QED Part 2

Figure 2. MU's Solution to a Problem.

4. Conclusion

MU shows that it is possible to model the learning of a very large and open domain. The solution is to use an arbitrary set of features built from the environment for initial encoding, and let experience dictate how the features should be revised to improve performance.

References

[Ammon, 1992] Ammon, K. Automatic Proofs in Mathematical Logic and Analysis. In *Proceedings of 11th International Conference on Automated Deduction.*

[Furse, 1990] Furse E. A Formal Expression Language for Pure Mathematics, Technical Report CS-90-2, Dept. of Computer Studies, The University of Glamorgan.

[Furse, 1993] Furse E. Escaping from the Box. In *Prospects for Intelligence: Proceedings of AISB93,* (Eds) Aaron Sloman, David Hogg, Glynn Humphreys, Allan Ramsey, Derek Partridge, IOS Press, Amsterdam.

[Furse, 1994] Furse E. The Mathematics Understander. In *Artificial Intelligence in Mathematics,* (eds) J H Johnson,S.McKee, A. Vella, Clarendon Press, Oxford.

[Morgan et a!., 1995] Morgan G., Furse E., and Nicolson R.I, Learning Problem Solving Heuristics From Worked Examples, *First European Cognitive Science Conference,* INRIA, France.