A RULE - BASFD TASK GFNFRATION SYSTFM

D. H. Sleeman

University ot leeds. U.K.
(Visiting Scientist. Oept of Computer Science. CMU, session 1980/1)

ABSTRACT

Considerable attention has been given to creating hypotheses which cover a set of instances. However, todate the related problem of generating a task to refine the hypothesis set has so far received very little attention. A task-generator has been implemented for a rule-based modelling system. LMS, Leeds Modelling System. Here we discuss how an analogous task-generator could be used with systems which do not constrain their hypothesis-space as strongly as LMS.

1. INTRODUCTION

Considerable attention has been given to creating hypotheses which cover a set of instances. [1], [2], [3], [4], [6], [7], [8] and [10]. However, todate the related problem *of* generating a task to refine the hypotheses set has so far received very little attention. Mitchell [7] is amoung the few people to address this Issue, and indicates how a problem generator would fit within the Version Space approach. Most workers have made the same basic assumption that we have made 1n our earlier Modelling work [10]. namely that the set of examples provided should be sufficiently rich to discriminate between all likely hypotheses. Obviously this assumption-becomes less viable as the number of potential hypotheses increases.

This task generator has been created as part of a system which models student's problem solving. LMS, Leeds Modelling System, presents the student with a task. 1s presented with the student's response and from this, hypothesizes models to explain the student's performance. (Indeed. LMS.represents each student model as an ordered production set; ordered. in the sense that the first rule which is satisfied, fires. See figure 2 for TRACES of typical models). In our work we have paid particular attention to the size of the search space and have shown that by concentrating on particular rule(s) it is possible to achieve very major reductions in the size of search space(s). In an earlier paper we discussed an algorithm for the systematic search of a hypothesis-space [12], and the need to create for each domain-rule a discriminatory problem-type, or Problem Template. Here we show how these techniques can also be applied to systems where the search has not been so systematic. The task-generator described here is for the Algebra domain, but we beleive that the approach outlined has considerable generality.

Indued, we claim that the Task Generator described here is suitable for domains in which the objective is to transform an initial state into either a pre-specified (goal) Slate or at least one with specified characteristics. (Generators for other task-types, for example the tasks described by Klahr, [5] and Siegler, [9], will be discussed in a subsequent paper). Further, it has been assumed that the domain knowledge has been formulated as a set of C0NDITION-ACTI0N rules; that the tasks will be processed using the same matching execution cycle as used here, namely the first satisfied rule fires.

2., The Alafiiina Domain.

Figure la gives a styalized form of the rules used to evaluate/solve Algebra tasks, and figure lb gives the set of associated mal-rules which have been noted in an earlier analysis of protocols, and used in a Modelling experiment, [11]. Styalized rules have been used for Task Generation as they capture all the essential issues, but avoid some of the messier detail. (Clearly, it would be possible to implement an algorithm to generate the 'real' rules from the styalized ones). Figure 2 gives 'IRACFs' of pairs of correct and 'buggy' models solving typical problems, and hence indicates the types of discriminatory tasks which the task generation algorithm must create.

3. Rule-based task generator:.

This analysis, further assumes that the model- and task-generation algorithms make use of the following classification for the CONDITION part of the rules:

a) Potential Interaction between Rules

Suppose the conditions of Rule 1 is Cl C2 ... Cm Cn and the conditions of Rule 2 is Cm Cn Cs .... Cz. then both Rules would be able to fire, if the problem presented contained a pattern of the form.-

Cl ... Cm Cn Cs ... Cz

As we are dealing with ordered production sets, the order of the rules determines which fires (first). And hence such a pattern would potentially discriminate between the models (...Rl...R2...) and (...R2...Rl...). [Whether or not the example does discriminate depends on the actions of the 2 rules].

An example of such a pair of rules it

ADDSUB	NUM +/- NUM
MULT	NUM ♦ NUM

Figure 1

a) RULES for the ALGEBRA domain (evaluative form and slightly styalized)

	LEVEL	PATTERN	'ACTION'
FIN2	1	(SHD X = M/N)	(SHD (M N)) or (SHD evaluated)
SOLVE	2	(SHD M * X = N)	(SHD X = N/M) or (SHD INFINITY)
ADDSUB	3	(M +/- N)	(evaluated)
MULT	4	(M * N)	(evaluated)
XADDSUB	5	(M*X +/- N*X)	((M +/- N) * X)
NTORHS	6	(lhs +/- M = rhs)	(lhs = rhs -/+ M)
REARRANGE	7	(+/-M +/- N*X)	(+/- N*X +/- M)
XTOLHS	8	(lhs = +/- M*X rhs)	(lhs -/+ M*X = rhs)
BRA1	9	(< N >)	(N)
BRA2	10	(M*<N*X +/- P>)	(M*N*X +/- M*P)

Where M, N and P are integers and where lhs, rhs etc are general patterns (which may be null), where +/- means either + or - may occur and where SHD indicates the String Head.
[Thus the state:

2*X + 5 + 6 = 9

could be matched by both the ADDSUB and NTORHS rules; which rule would fire, depends on the relative order of the rules within the model].

b) Some MAL-RULES for the Domain

	LEVEL	PATTERN	'ACTION'
MSOLVE	2	(SHD M*X = N)	(SHD X = M/N) or (SHD INFINITY)
MNTORHS	6	(lhs +/- M = rhs)	(lhs = rhs +/- M)
M2NTORHS	6	(lhs1 +/- M lhs2 = rhs)	(lhs1 +/- lhs2 = rhs -/+ M)
M3NTORHS	6	(lhs1 +/- M lhs2 - rhs)	(lhs1 +/- lhs2 = rhs +/- M)
MXTOLHS	8	(lhs = +/- M*X rhs)	(lhs +/- M*X = rhs)
M1BRA2	10	(M *<N*X +/- P>)	(M*N*X +/- P)
M2BRA2	10	(M*< N*X +/- P>)	(M*N*X +/- M +/- P)

Using the same conventions as above.

Figure 2

Pairs of correct and 'buggy' models executing typical tasks.

a) shows (MULT ADDSUB SOLVE FIN2) and (ADDSUB MULT SOLVE FIN2) solving 2X = 3 * 4 + 5.

[The first line gives the initial state and all subsequent lines give the rule which fires and the resulting state].

	2X = 3 * 4 + 5		2X = 3 * 4 + 5
MULT	2X = 12 + 5	ADDSUB	2X = 3 * 9
ADDSUB	2X = 17	MULT	2X = 27
SOLVE	X = 17/2	SOLVE	X = 27/2
FIN2	(17/2)	FIN2	(27/2)

b) shows (NTORHS ADDSUB SOLVE FIN2) and (MNTORHS ADDSUB SOLVE FIN2) solving 2X + 5 = 9.

	2X + 5 = 9		2X + 5 = 9
NTORHS	2X = 9 - 5	MNTORHS	2X = 9 + 5
ADDSUB	2X = 4	ADDSUB	2X = 14
SOLVE	X = 2	SOLVE	X = 7
FIN2	(2)	FIN2	(7)

The discriminatory patterns, which we shall also refer to as Overlap Patterns, in this case being:

NUM +/- NUM * NUM
and NUM * NUM +/- NUM

b) Non-interacting or Order-Independent rules Rules

Non- interacting rules are ones which whatever: the pattern would not be in conflict, ie the patterns do NOT overlap. For example, if the conditions of RI are CI C2 C3 and of R2 are CI C2 C4 then these would NOT compete.

c) Subsume- Rule RI is said to subsume R2. if RI's condition set is an ordered sub-set(l) of R?'s conditions (the effect of RI being placed before R2. in an ordered Production Set. is that the PS would act as if R2 did not occur (2)). An example of a subsuming/subsumed pair from this domain is ADDSUB and REARRANGE.

Note that Subsuinption and 'Non Interaction' can he decided by inspection of the CONDITION parts of the rules; potential interaction/conflict depends critically on the form of the problem considered, and so has to be decided with respect to each problem type.

In older to generate a type of problem which would allow a rule to file we need to use the rules in the opposite direction to which they are used to solve a problem. That is given the MULT rule: -

(STR1 NUM STR2) => (STR1 NUM • NUM STR2)

the algorithm checks if NUM is contained in the input string, and if so replaces it by the string NUM ♦ NUM, (STRi are strings which can be null). To initiate the generation algorithm we need to specify an initial string, a set of rules which can be used in generation mode and a terminating condition (number of iterations, rules which the final pattern must activate etc.). Figure 3 shows the performance of the system when given the initial state, (SHI) NUM). Rule Net (MUI SOLVF FIN2) and the critereon that MU1 T and then SOI VF must be activated.

3.1 The *algorithms* to generate a problem to dl£_crj.mina_Le_ between to models

The basis of this approach is the discrimination between 2 models. However, if one has N models then it would be sensible to choose a feature, say the presence/absence of a particular rule which, as nearly as possible, splits the hypothesis space into ?, and to use the algorithm outlined in this section with 2 models, one of which has the chosen feature and *one* that does not.

Note that if one is systematically exploring a rule-space, as do both versions of IMS. [10], [12], then the only variants encountered should be rule/mal-rules and the ordering of the rules. However, what follows is a more general analysis which is needed if the space is not being searched systematically. In general, there are *2* cases which need to be distinguished between namely:

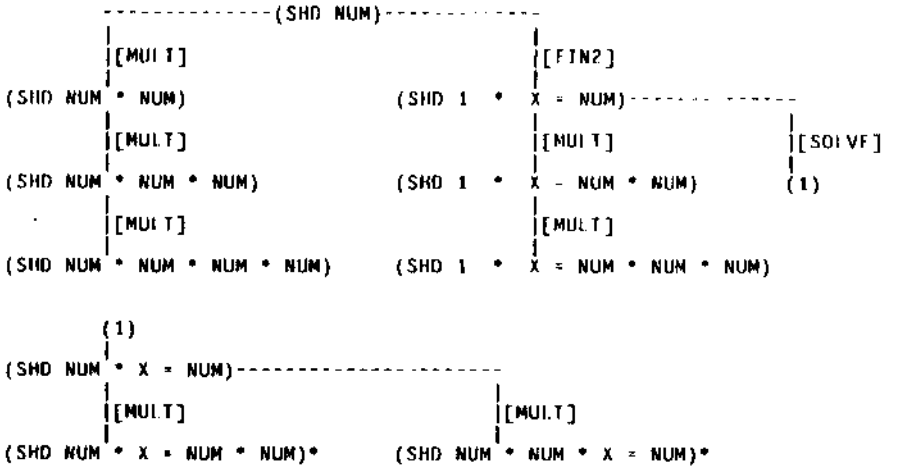
A. Model, which have the some rules but differently ordered

Initially, we need to distinguish in each model, the sub set of rules which can potentially interact with other rules. Suppose that the 2 models were:

(FIN2 SOIVE ADOSUB MUI T)
(MUIT ADDSUB SOIVF FIN2)

Figure 3

Generation of a problem for (MUI T SOI VF FIN2), starting from the basic form (SHD NUM).



The Names On The Arcs show The Rules Used To Create The Next String. The starred configurations are returned by the system as ones which meet the critereon specified. (NB a slightly simplified version of FIN2 has been used).

Then the 2 rules which could interact are ADDSUB and MULT; these rules are ordered differently in these Models and so the algorithm would attempt to generate a problem which discriminates between the 2 orderings of the rules, MULT and ADDSUB. (If they are not ordered differently then a discrimination is not possible; eg (TIN? SOIVE ADDSUB MULT) and (ADDSUB MULTI FIN2 SOIVL) are equivalent models).

The initial algorithm to create such a Template attempts to create a task which activates al 1 the rules contained in the models and which contains the Overlap Pattern, OPij. In effect, the models being considered contain just one additional rule to the model considered in figure 3, and so we can anticipate the form of the resultant tree. Such a search is essentially breadth first, and so as at each level several rules can be applied the number of active nodes increases rapidly. A much 'tidier' approach is to produce Problem Templates for the 'basic' models, that is for:

(ADDSUB SOI Vf FIN2)

(MUI r soi vh HN?)

and then merely to extend these Problem templates to 'accommodate' the additional rule, le MUI / or ADUSUH; in figure 4 we list the Pfs generated for this domain's rules and associated ma 1 rules. (In an earlier paper. [12], we discuss the creation of such PTs. as well as the generation of PTs to discriminate between interacting pairs of rules eg ADDSUB and M2N10HHS).

The revised algorithm, to discriminate between 2 models, calculates the Overlap Pattern(s) and attempts to 'extend' the Problem Template for rule), PTi, by using Rj in generative mode. The template is accepted if it contains OPij and if it satisfies Ri and Rj. Similarly, the algorithm attempts to extend PTj using Ri in generative mode; the analogous tests are applied. For this example, as we have noted, the Overlap Patterns are:

NUM +/- NUM * NUM

NUM * NUM +/- NUM

and the discriminatory tasks created by extending the PTs for the ADDSUB and the MUI T rules are:

NUM * X = NUM * NUM +/- NUM

NUM * X - NUM +/- NUM * NUM

[In general there may be M pairs of wrongly ordered rules. in which case the discriminating problem should satisfy the above condition for each pair. In practice, one might prefer to generate a problem to discriminate between a single pair, and to work with the 'refined' models subsequently].

B. Models which have_have differing rules.

This algorithm considers rules which are known not to interact. There are 2 basic cases to be considered here, namely:

a) where the CONDITIONS of rules. Ri and Rj. are different, (3).

b) where the ACTIONS of the rule-pair are different.

The algorithm treats rules which differ in both their CONDITION and ACTION parts, as if only their CONDITION parts differ (as it is easier to verify discrimination has been achieved in this case). [Rules which have identical CONDITION and ACTION parts should NOT occur and are reported as data errors]. For the moment, we shall assume there is only one rule pair to be discriminated between in the two models (this approach may be generalized later).

The 2 sub-algorithms are:

Different Condition Part

If we use the notation that C(Ri) is the condition part of the ith rule and PTi its Problem Template, then:

If C(R1) is not satisfied by PT2 then PT2 di scr iminates.

Similarly, if C(R2) is not satisfied by PT1 then PT1 discriminates.

If both of the above tests fail, then check that the actions of the 2 rules differ; if they are the same, report an error as discrimination will not be achieved with the current PTs. [Subsequently, we plan to implement an algorithm which will extend the PTs to ensure discrimination in such cases].

Ligure 4

Styalized version of Problem templates for the Domain Rules.
(NB the mal-rules use the PT of the 'parent' rule.)

FIN2	(1 * X - NUM // NUM)
SOIVE	(NUM * X - NUM)
ADDSUB	(NUM * X = NUM +/- NUM)
MUI T	(NUM * X - NUM * NUM)
XADDSUB	(NUM * X +/- NUM * X = +/ NUM)
NTORHS	(+/ NUM * X +/ NUM - +/ NUM)
XIOI HS	(NUM * X = - NUM - NUM * X)
RfARRANGE	(NUM +/ NUM * X = NUM)
BRA1	(NUM * X - NUM * < NUM >)
BRA2	(NUM * X = NUM * < NUM * X +/- NUM >)

Note the PTs returned by the algorithm do NOT contain SHD; however, this is present in the intermediary forms so that the expressions can be symbollically evaluated by FIN2 and SOLVE.

Thus this problem reduces to that of ensuring that appropriate Templates have been provided as data or subsequently 'grown'.

Different Action Part

Rules which have the same CONDITION-part but differing ACTION parts are assumed to be distinguished by the differing actions, (4). Thus for NIORHS/MNIORHS it is assumed that problems associated with NIORHS would give different results with MNIORHS. (And so the problem template for NIORHS does not need extension to cover this mal-rule; also see figure 2).

3.2 Summary of Cases

If the models to be distinguished between, are such that they fall within both class A and B, then at present the algorithm considers case B before case A. Subsequently we may upgrade the algorithm so that both types can be considered simultaneously; although we have some reservations about the use of such 'complex' examples.

To summarize if we have models:

(ADDSUB SOLVE FIN2)
(XADDSUB SOLVE FIN2)

this would be classified as an instance of the Ba algorithm.

(NIORHS ADDSUB SOLVE FIN2)
(MNIORHS FIN2 SOLVE ADDSUB)

would be classified as an instance of the Bb algorithm.

(NIORHS ADDSUB SOLVE FIN2)
(SOLVE FIN2 M2NIORHS ADDSUB)

would be classified as an instance of Ba and A as M2NIORHS and ADDSUB potentially interact.

Whereas

(MULT NIORHS ADDSUB SOLVE FIN2)
(ADDSUB MNIORHS MULT SOLVE FIN2)

is an example which falls into classes A and B (lib), as MULT and ADDSUB potentially interact and because NIORHS and MNIORHS have different actions.

The other assumption which we have made so far is that no additional rules have to be included in the rule-set in order to generate a discriminatory problem. It is possible to envisage situations, where this is not a valid assumption, but we have been happy to make this assumption in order to 'contain' what is potentially an enormous search space. However, it is only necessary to add additional rules to the generation set if the initial search fails. (In the limit all the domain rules should be contained in the 'generative' rule set).

3.3 Templete Instantiator.

So far we have not implemented a Template Instantiator for this domain; this is not a very demanding task, but neither is it a very rewarding one as similar implementations have been done many times before. However, such an implementation is needm! to take full advantage of any on line model liny capability.

Footnotes

1. Cl C2 subsumes Cl C2 C3 whereas C2 Cl does not.

2. Note carefully this use of SUBSUME. We are using SUBSUME only to discuss ruleCONDITIONs.

3. In order to make a comparison between different rules (mal-rules), it may be necessary to introduce a standardized notation for the variables. Eg the first parameter becoming $p1 etc.

4. For the moment, we are ignoring numerical degeneracy which sometimes occurs; indeed, with a large set of models it is difficult to choose problems to avoid it completely. For the present, our only solution is to bring such degeneracy to the investigator's attention when the models evaluate specific problems.

Acknowledgements

Jaime Carbonell, Pat langley. N.S. Sridharan and K.R. James (leeds) for helpful comments on earlier drafts.

To leeds for granting Sabbatical leave which has enabled me to concentrate on this work; and to CMU for providing facilities and a very stimulating enviroment in which to continue it.

REFERENCES

1. Buchanan, B.G. 1974. Scientific Theory formation by computer, in Proceedings of the NATO Advanced Study Institute on Computer Orientated Learning Processes.

2. Burton, R.R. 1981. Diagnosing bugs in a simple procedural skill. In Intelligent Tutoring Systems, ed. D. Sleeman and J.S. Brown, New York: Academic Press.

3. Cohen, B.I. & Sammut, C.A. 1978. CONFUCIUS: A Structural Concept Learning System, Australian Computer Journal, 10, pp 138-143.

4. Hayes-Roth, F. & McDermott, J. 1978. An inference matching techniques for Inductive Abstractions. CACM. 21, pp401-410.

5. Klahr, D. and Wallace, J.G. 1970. The development of serial completion Strategies. British Journal of Psychology, pp 243-257.

6. Michalski, R. (1979). Pattern Recognition as Rule-Guided Inductive Inference, IEEE Transactions on Pattern Analysis and Machine Intelligence, pp ??

7. Mitchell. T.M. 1978. Version Spaces: an approach to Concept learning. Ph.D. Thesis, Dept of Computer Science, Stanford.

8. Quinlan, J.R. 1979. Discovering Rules by Induction from large collections of Examples. In (ed) D.Michie, Expert systems. in the Micro.:electronic age.. Edinburgh: FUP, pp 168-201.

9. Siegler, R.S. 1978. The Origins of Scientific Reasoning, in Childrens ThinKing; what develops. ed R.S. Siegler. Hillsdale, NJ: LEA.

10. Sleeman, D.H. and Smith. M.J. 1981. Modelling Student's Problem Solving, AI Journal. pp 171-187.

11. Sleeman, D.H. 1981. Assessing aspects of competence in Basic Algebra. In Intellegent •Tutoring Systems.. ed. D.Sleeman and J.S. Brown, New York: Academic Press.

12. Sleeman, D.H. (to appear). A Rule-based Modelling system.