Pre-publication draft of a paper which appeared in the Proceedings of the Ninth Australian Joint Conference in Artificial Intelligence (AI'97), Perth, Australia. Berlin: Springer, pp 349-358

Using Decision Trees for Agent Modelling: A Study on Resolving Conflicting Predictions

Bark Cheung Chiu, Geoffrey I. Webb, and Zijian Zheng
School of Computing and Mathematics, Deakin University, Australia

Abstract

Input-Output Agent Modelling (IOAM) is an approach to modelling an agent in terms of relationships between the inputs and outputs of the cognitive system. This approach, together with a leading inductive learning algorithm, C4.5, has been adopted to build a subtraction skill modeller, C4.5-IOAM. It models agents' competencies with a set of decision trees. C4.5-IOAM makes no prediction when predictions from different decision trees are contradictory. This paper proposes three techniques for resolving such situations. Two techniques involve selecting the more reliable prediction from a set of competing predictions using a free quality measure and a leaf quality measure. The other technique merges multiple decision trees into a single tree. This has the additional advantage of producing more comprehensible models. Experimental results, in the domain of modelling elementary subtraction skills, showed that the tree quality and the leaf quality of a decision path provided valuable references for resolving contradicting predictions and a single tree model representation performed nearly equally well to the multi-tree model representation.

1 Introduction

Inductive learning techniques provide approaches to building agent models in the form of knowledge acquisition by observing the agents' behaviours. For example, they can be used in an intelligent tutoring system (ITS) to model a student's competencies. They make hypotheses about a student's knowledge of that problem domain. They learn theories, from examples, like many people do. Decision tree learning or rule learning provides the additional advantage of representing the acquired knowledge in explicit form such that people can interpret it.

The use of inductive learning for agent modelling has been studied previously, eg. (Desmoulins and Van Labeke, 1996; Gilmore and Self, 1988). An induction based modelling system may require prohibitive resources for implementation if its inductive engine is tightly linked to the cognitive aspects of an agent. An Input-Output Agent Modelling (IOAM) approach allows a system to treat the operation of the cognitive system as a black box and models the operation in terms of the relationships between the inputs and outputs of the system. Therefore, a general-purpose classifier learning algorithm can be employed as an induction engine. Examples of modelling systems which employ the IOAM approach include Feature Based Modelling (FBM) (Webb and Kuzmycz, 1996), Relational Based Modelling (Kuzmycz, 1995), FFOIL-IOAM and C4.5-IOAM (Chin et al., 1997). Note that they use different induction engines. C4.5-IOAM uses C4.5 (Quinlan, 1993), a well- known and general-purpose learning algorithm, as its induction engine. It has been used to model students' competencies of elementary subtraction skills with a set of decision trees. Comparative evaluations of C4.5-IOAM against FBM (Webb et al., 1997) and FFOIL-IOAM (Chin et al., 1997) have shown that the use of C4.5 increased the number of predictions made without significantly altering the accuracy of those predictions.

However, C4.5-IOAM makes no prediction when the individual predictions from different decision trees are contradictory. If C4.5-IOAM is augmented by a mechanism for resolving conflicting predictions, it might make more predictions without affecting the prediction accuracy. Three techniques have been explored for this purpose. Two of them involve selecting the more reliable prediction from a set of competing predictions. The other technique merges multiple decision trees into a single tree. This approach offers the additional advantage of producing more comprehensible models. This paper presents an empirical study of how conflicting predictions can be resolved with these alternative techniques.

2 An overview of C4.5-IOAM Subtraction Modeller

The C4.5-IOAM subtraction modeller manipulates an n-digit subtraction problem by treating it as n separate column problems. In this subtraction modeller, context features and action features, adopted from the FBM (Kuzmycz and Webb, 1992), are used to represent inputs and outputs. Context features describe the problems with which a student is faced. Action features describe aspects of a student's actions for a particular problem. There are eleven action features:

Result=M-S, Result=M-S-l, Result=l0+M-S, Result=10+M-S-1, Result=M, Result=S, Result=zero, Result=M-S 2, Result=l0+M-S-2, Result=S-M and Result=correct, where M and S stand for minuend and subtrahend digits respectively. Action features are not mutually exclusive. That is, a student's action may correspond to more than one action feature. However, C4.5 requires mutually exclusive classes. Thus, C4.5-IOAM uses eleven decision trees to model different aspects of a student's actions (behaviour).

The context features of a unit problem are described by 12 attributes. The first four attributes, *M_is_0, S_is_0, S_is_9* and *S_is_BK* (BK stands for blank), are self-explanatory. The rest of them are listed below with their meanings where N stands for Not Available.

- *M_vs_ S*: {G, L, E}, *M* is greater (G) or less than (L), or equal (E) to *S*.
- *M_L_is_0*: {T, F, N}, *M* in the column to the left is zero.
- *M_L_is_1*: {T, F, N), *M* in the column to the left is one.
- *M_R_is_0*: {T, F, N}, *M* in the column to the right is zero.
- *S_R_is_9*: {T, F, N} *S* in the column to the right is nine.
- *M_S_R*: {G, L, E, N}, similar to *M_vs_S*, but it describes the column to the right.
- *M_S_2R*: {G, L, E, N}, similar to *M_vs_S*, but it describes two columns to the right.
- *Column*: {L, I, R}, the current column is left-most (L), inner (I), or right-most (R).

Figure 1 illustrates how 11 training examples, one for each decision tree, are formed from one single column problem. The context features, described by 12 attributes, are extracted based on the problem environment and applied to each example. At the inner column, *M* (minuend digit) is nine, *S* (subtrahend digit) is zero, and the student's answer is nine. Two action features, *Result=M-S*, and *Result=M*, correspond to the student's action. These two action features are, therefore, set as T. The other action features are set as F. One 3-digit subtraction problem will generate three training examples for each decision tree. After all examples of a student's subtraction performance are processed, C4.5 is used to infer a decision tree for each training set.

When C4.5-IOAM predicts a student's answer for a problem, the problem context is extracted and used to consult the eleven decision trees. The decision trees being consulted are then confined to those - each predicts the presence of the corresponding action. If these predictions lead to the same digit, the system adopts the digit as the final prediction. Otherwise, the system makes no prediction about the student's answer.

Figure 2 shows a sample theory inferred by C4.5-IOAM. Decision trees with only one leaf labelled F predict the student will not exhibit the corresponding actions. *Tree_M* predicts that if the subtrahend digit is zero, the student will assign the minuend as the answer.

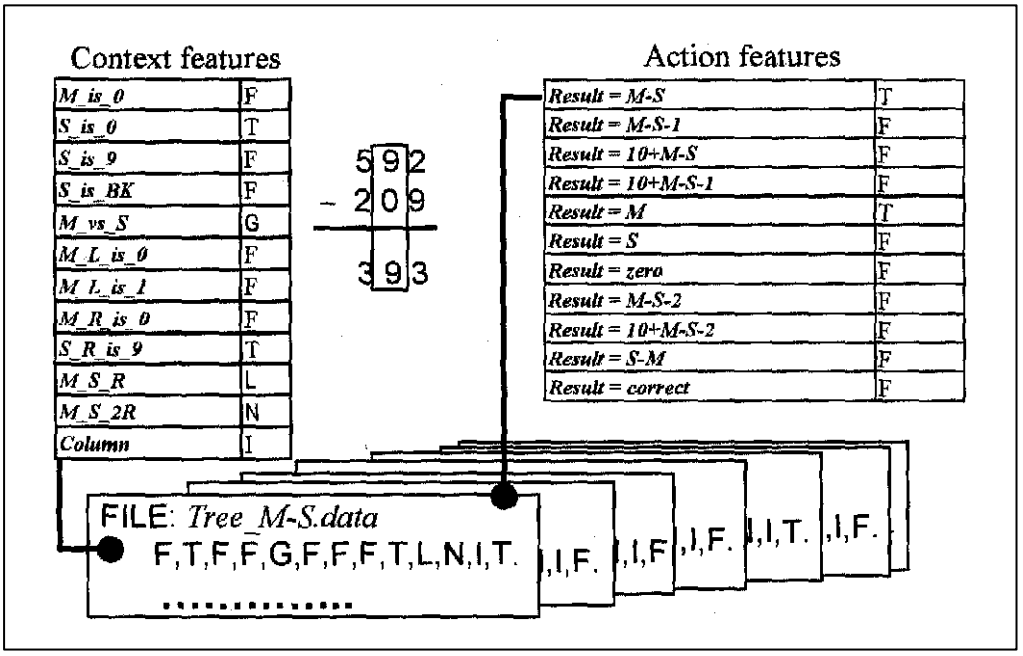

Figure 1: Formation of a column's training examples for decision trees

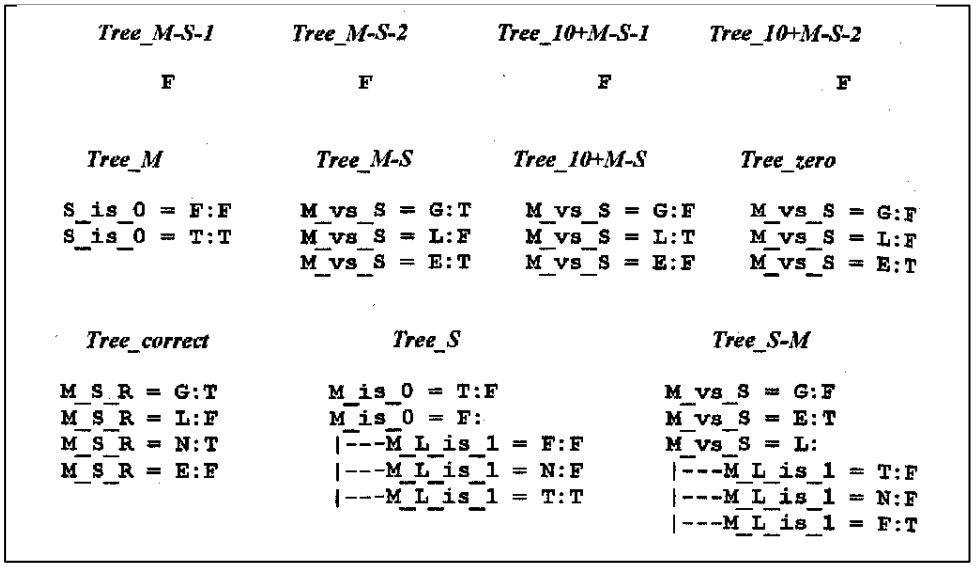

Figure 2: A theory inferred by C4.5-IOAM

3 Techniques for resolving conflicting predictions

The current version of C4.5-IOAM makes no prediction whenever different decision trees make conflicting predictions. The following conflict-resolving techniques are proposed to improve the prediction rate of the current system.

3.1 Using quality measure of a decision tree

The prediction rate of the system can be improved by adopting the predictions from the more reliable decision trees. A conflict resolution technique like voting is not suitable because the decision trees predict different aspects of a student's actions. We assume decision trees making conflicting predictions have different characteristics in terms of measurable quality such as prediction accuracy and prediction error rate.

We employed stratified ten-fold cross-validation (Kohavi, 1995) for estimating the error rate of each decision tree. For each action feature, the training examples are randomly divided into ten equal-sized partitions. Each partition, which preserves the original class distribution, is used in turn as test data for the decision tree trained on the remaining nine partitions. The total numbers of correct and incorrect predictions of these tests are then used to estimate the error rate of the decision tree trained on the whole training set. A C4.5-IOAM system can improve its prediction rate by associating decision trees with estimated error rates, and consulting the trees in a ranked order. With this consulting order, the first tree that gives a positive prediction is used to make the system's prediction. This method contrasts to the current system, which consults the trees in parallel.

3.2 Using a leaf quality measure

The error rate of a tree reflects the overall quality of the tree. We know that different leaves have different predictive power because the evidence on which they make predictions are different. A leaf with less support on a high quality tree may make a poorer prediction than a leaf with more support on a tree with a relatively lower quality. A closer look at how C4.5 builds a decision tree may help to explain this. C4 .5 builds decision trees by using a divide and conquer strategy. It recursively selects the best attribute, which will generate less complex subtrees, to form branching nodes. Examples of the training set are partitioned based on the selected attribute. This process continues until the training set at a node cannot be further divided, for example, because no significant split exists. In the case that a terminal node (leaf) contains examples of different classes, it takes the majority class as the leaf label. When the tree is used to predict the class of an unseen (test) example, it traverses the unseen example through a decision path and suggests the leaf label of the path as the prediction. The reliability of this prediction can be estimated by examining the distribution of the classes of training examples at the leaf node. We are only concerned with the leaf labelled with T for predicting that a student will exhibit a particular action. Let t denote the total number of examples, and e denote the number of examples labelled with F, at the leaf node with label T, a value based on the Laplace formula, $(e + 1)/(t + 2)$, can be used as the estimated error rate of a leaf.. A C4.5-IOAM system can therefore adopt the prediction of a decision tree of which the leaf node of the decision path is associated with a lower error measure.

3.3 Using a single tree instead of multiple trees

The tree and leaf quality approaches attempt to resolve multiple predictions. Another approach is to circumvent the problem by producing only one prediction. We can achieve this by developing a single tree that predicts the most useful action feature for predicting an agent's actions in a given context. Such a tree requires a training set labelled with the most useful action feature for each example. We propose a two- phase identification algorithm (see Figure 3) which can be employed at the training stage. For each training example which is accompanied with more than one action feature, each action feature is validated by a lazy Bayesian tree[1] (Zheng and Webb, 1997) trained from all other training examples. The lazy Bayesian tree is used for the sake of computational efficiency. This filtering process reduces the number of examples with multiple action features. At the second stage, those training examples with multiple action features form a temporary test set. A temporary decision tree, trained on examples that each has a unique action feature, predicts the most useful action feature for each example in the test set. The ultimate training set, in which each example is labelled a most useful action feature, or as unknown if a most useful action feature cannot be identified, infers a single tree for the system.

[1] For each test example, the Lazy Bayesian Tree learning algorithm generates one relevant decision path. The leaf of the path uses a local naive-Bayesian classifier, instead of a majority class, to classify the test example.

Given: raw training set M with N examples, from a single student, in the form of
(Att, undetermined_action) where Att is a set of problem context features;
11 training sets, A_i *, each has N examples in the form of (Att,* a_i *),* $a_i \in \{F,T\}$ *where*
F/T stands for the status, absence/presence, of a corresponding action feature.
Output a training set of examples in the form of (Att, Action), Action $\in$ *{M-S, M-S-1,…,correct, unknown}.*

FOR n: =1 to N DO
obtain a status list ($a_1,\ldots,a_i,\ldots,a_{11}$*) from the n-th examples from training sets* $A_1 \ldots A_i \ldots A_{11}$
generate an index list L, of competing actions, where L = (.,i,..) for any $a_i = T$
IF L has more then one element THEN L = Reduce_competing_actions (L, n)
IF L has one element i THEN undetermined_action: = $Action_i$
ELSE IF L is empty THEN undetermined_action: = unknown
ELSE
append $example_n$ *to undetermined example list U; and*
append L to a list of competing action lists LL

FOR each undetermined example, $example_u$ *, in U DO*
retrieve the corresponding L from LL
undetermined_action = Classify_within_competing_actions($example_u$ *, L)*

*Process name: Reduce_competing_actions. /*Phase 1: Internal identification */*
Given: L, an index list of competing actions; n, an index of the current example.
Output L, a revised index list of competing actions.

FOR each index i in L DO
build a lazy Bayesian tree LBT_i *based on* A_i *excluding the example* $_n$
IF the predicted class of the test item, $example_n$ *, is F*
THEN remove i from L
RETURN L

*Process name: Classify_within_competing_actions. /*Phase 2: Global identification*/*
Given: $example_u$ *, a test example ; L, a list of index of competing actions.*
Output: an action, or unknown.

prepare a temporary training set by copying all examples, for which class labels match the actions described by the index in L, from the raw training set
IF the temporary training set is not empty
THEN
build a decision tree D to test $example_u$ *and RETURN the action predicted by D*
ELSE
RETURN unknown

Figure 3. Two-phase identification process.

4 Experiments

The same data set, which has been used to evaluate C4.5-IOAM and other IOAM based subtraction modellers (Webb et al., 1997; Chiu et al., 1997), was used to evaluate experimentally the techniques discussed in Section 3. The data came from 73 primary school students who were administered with five rounds of subtraction-problem tests. For each student, a modelling system used all data from prior rounds to build a student model and used the current round data to test the student model. The data set allows a system to conduct 264 training-testing processes.

The performance of the current version of C4.5-IOAM was used as a baseline. There were a total of 30,474 student answers, of which 3,630 were incorrect answers. The C4.5-IOAM system made 28,700 predictions, of which 26,507 (92%) were correct. Of the system's 1,999 predictions that a subject would provide an incorrect digit for a column, 1,347 (67%) were accurate, predicting the exact digit provided. We used the symbols +Tree_QTY, +Leaf_QTY and Single-tree to represent new versions that were implemented by introducing quality measures on trees and leafs, and merging multiple trees to a single tree respectively. The performance of these new versions was compared with C4.5-IOAM. Table 1 summarises these results.

We used a two-tailed pair-wise t-test to evaluate the statistical significance of the observed differences. The performance differences of the 264 model tests are summarised in Table 2, where the number at the intersection of row A > B and column C represents the number of cases in which system A outperformed system B for the performance category C.

All new versions achieved significant improvement in prediction rate while their overall prediction accuracy dropped slightly. However, their numbers of correct predictions were still higher than that of C4.5-IOAM. The introduction of quality measures for decision trees and leaf nodes increased the number of error predication with an expanse of error prediction accuracy. Again, their numbers of correct predictions were still higher than that of C4.5-IOAM. In this aspect, the difference between the Single-tree version and C4.5-IOAM was not significant.

Regarding the inferred theories generated by these systems, only the Single-tree version generated a single tree for describing a student's problem solving competency. Figure 4 and Figure 5 show the outputs of the single-tree and of the multi-tree versions when a student's first round performance was captured. Both models exhibited identical performance in predicting the student's next round answers. The multi-tree representation tells the absence and presence of each action in detail. For the Single-tree version, a leaf labelled as correct covers those actions leading to correct answers, while a leaf with other label tells how an erroneous action is predicted. This single model is likely to be easier for teachers and students to understand.

	C4.5-IOAM	+Leaf_QTY	+Tree_QTY	Single-tree
Number of predictions made	28.700	30.093	29.783	30,130
Prediction rate	94.2%	98.7%	97.7%	98.9%
Number of predictions that were correct	26,507	27,543	27,308	27,495
Prediction accuracy	92.4%	91.5%	91.7%	91.3%
Number of error predictions made	1,999	2,173	2,346	2,095
Prediction rate	55.1%	59.9%	64.6%	57.7%
Number of error predictions that were correct	1.347	1,426	1,485	1,373
Prediction accuracy	67.4%	65.6%	63.3%	65.5%

Table 1: Performance of new versions created by three kinds of treatment

	More predictions Made	Higher accuracy	More error predictions Made	Higher accuracy (for errors)
+Leaf_QTY> C4.5-IOAM	197	73	40	7
C4.5-IOAM >+Leaf_QTY	0	105	0	24
	(p<0.0001)	(p=0.0001)	(p<0.0001)	(p=0.0081)
+Tree_QTY> C4.5-IOAM	186	80	79	11
C4.5-IOAM >+Tree_QTY	16	103	1	35
	(p<0.0001)	(p=0.0001)	(p<0.0001)	(p=0.0029)
Single-tree> C4.5-IOAM	195	83	48	17
C4.5-IOAM >Single-tree	14	114	33	39
	(p<0.0001)	(p<0.0001)	(p=0.2045)	(p=0.1460)

Table 2: Observed differences in performance between new versions and C4.5-IOAM

Tree_actions

```
M_S_R = G: correct
M_S_R = N: correct
M_S_R = L:
|---M_vs_S = G: M-S
|---M_vs_S = L: 10+M-S
|---M_vs_S = E: S-M
M_S_R = E:
|---M_S_2R = G: correct
|---M_S_2R = L: M-S
|---M_S_2R = E: correct
|---M_S_2R = N: correct
```

Figure 4: Knowledge representation inferred by a single-tree modeller

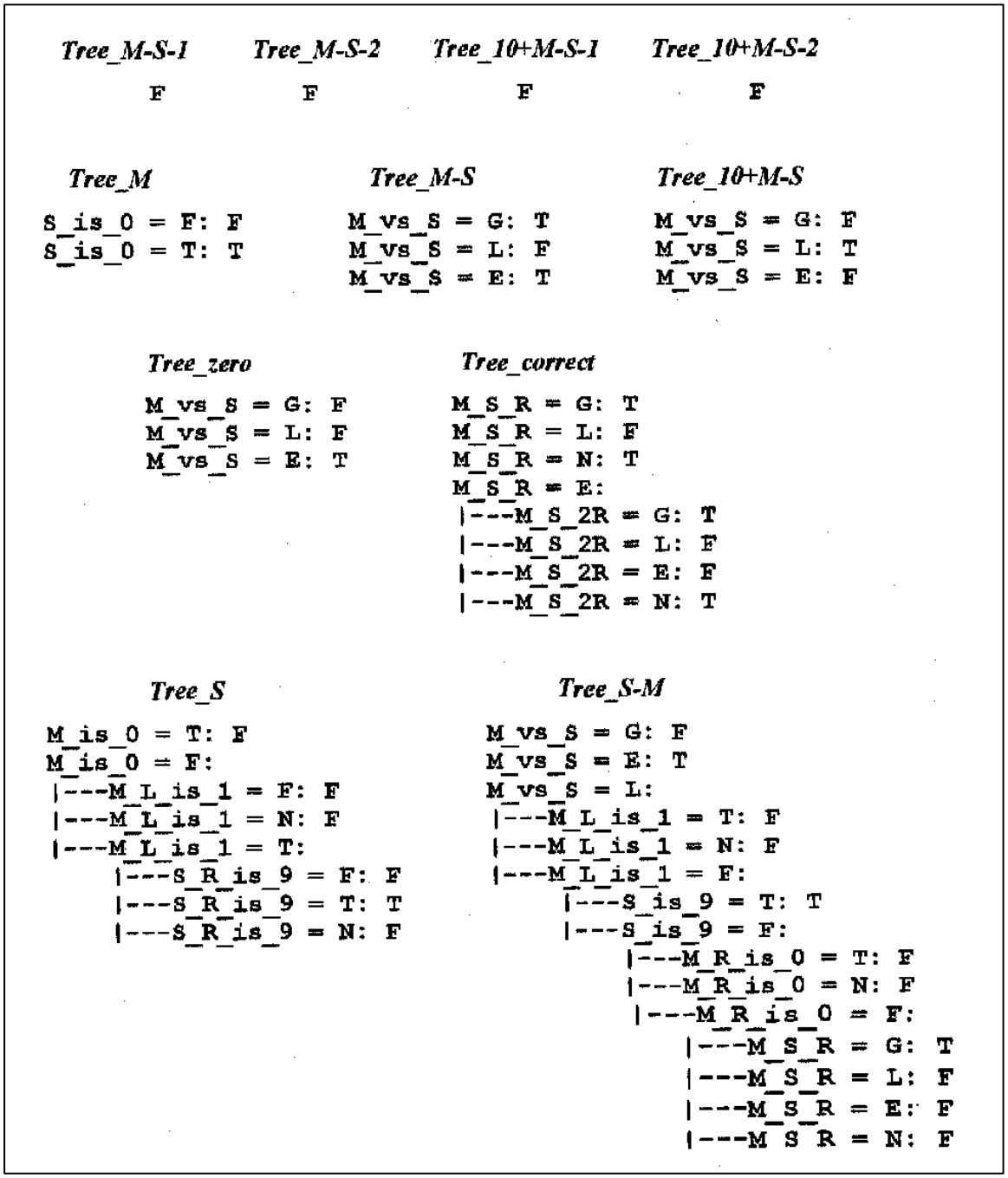

Figure 5: Knowledge representation inferred by a multi-tree modeller

5 Conclusions

The main tasks of modelling systems mentioned in this paper are learning to predict correct and erroneous actions, and generating theories describing agents' behaviours in subtraction problem solving. The main problem discussed in the paper is how to resolve conflicting predictions about an agent's action. We have described and evaluated three techniques for this objective.

Techniques of employing quality measures on decision trees and leaf nodes in resolving conflicting predictions at testing stage have been shown to be effective for this purpose. These two methods cover two aspects of resolving conflicts: adopting a decision from a point of view at global level; and considering the judgement based on local experience. It is quite similar to consulting human experts. While an engineer might have good abstract knowledge but lack sufficient experience for a particular case, an ordinary person could have encountered numerous similar examples and could be an expert for that case. The employment of merging multiple trees into a single tree shifts the conflict resolution to the training stage. The results are also promising. The Single-tree version achieved significant improvement in prediction rate with a slight drop in overall prediction accuracy. However, the number of correct predications was still higher than that of the original system. There was no significant performance difference in predicting incorrect answers, when it was compared with the baseline.

The use of multiple decision trees to represent different aspects of a student's action allows an ITS administrator to diagnose each action in detail. Yet, if a human tutor wants to get the whole picture of a class, thirty students, for example, a picture represented by thirty single trees should be preferable when compared with that involving hundreds of trees.

References

[1] Chin, B. C., Webb, G. I., and Kuzmycz, M. (1997). **A comparison of first-order and zeroth-order induction for Input-Output Agent Modelling.** In Jameson A., Paris C., and Tasso C., eds., *Proceedings of the Sixth International Conference on User Modeling, UM97*, 347-358.

[2] Desmoulins, C., and Van Labeke, N. (1996). **Towards student modelling in geometry with inductive logic programming**. In Brna, P., Paiva, A., and Self, J, eds., *Proceedings of the European Conference on Artificial Intelligence in Education.*

[3] Gilmore, D., and Se1f, J. (1988). **The application of machine learning to intelligent tutoring systems.** In Self, J., ed., *Artificial Intelligence and Human Learning: Intelligent Computer- aided Instruction*. London: Chapman and Hall. 179-196.

[4] Kohavi, R. (1995). **A study of cross-validation and bootstrap for accuracy estimation and model selection**. *Proceedings of 14th International Joint Conference on Artificial Intelligence*, 1137-1143.

[5] Kuzmycz, M, and Webb, G. I. (1992). **Evaluation of Feature Based Modelling in Subtraction**. *Proceedings of the Second International Conference in ITS, ITS92*. 269-276.

[6] Kuzmycz, M. (1994). **A dynamic vocabulary for student modelling**. *Proceedings of the Fourth International Conference on User Modelling*, 185-190.

[7] Quinlan, J.R. (1993). ***C4.5: Programs for Machine Learning***. San Mateo, CA: Morgan Kaufmann.

[8] Webb, G. I., Chiu, B. C., and Kuzmycz, M. (1997). **A comparative evaluation of the use of C4.5 and Feature Based Modelling a induction engines for Input Agent Modelling**. *International Journal of Artificial Intelligence in Education*.

[9] Webb, G. I., and Kuzmycz, M. (1996). **Feature Based Modelling: A methodology for producing coherent, dynamically changing models of agent's competencies**. *User Modelling and User-Adapted Interaction* 5(2): l17-150.

[10] Zheng, Z., and Webb, G. I. (1997). **Lazy Bayesian Tree**. *Technical Report TC97/07*, School of Computing and Mathematics, Deakin University.