# Chapter 2

# Co-operative inductive learning in a multi-agent learning system.

Recall from Chapter 1 that a CILT-MAL system consists of a number of learners, i.e. data mining techniques and human experts. The learners discussed here learned by means of induction to form sets of concept descriptions. The learners did not learn in isolation, but co-operated with one another in teams, hence they formed co-operative inductive learner teams. A co-operative inductive learner team can be modelled as a multi-agent learning system, with co-operative inductive learners participating as learning agents within this multi-agent learning environment.

This chapter is organised as follows. Section 2.1 introduces the concept of inductive learning and discusses co-operative inductive learner teams (CILT). This is followed by Section 2.2, which introduces a multi-agent learning system and the concept of an intelligent data analysis tool. Section 2.3 describes the way in which co-operative inductive learner teams can be modelled as a multi-agent learning system and hence describes the CILT-MAL system used for this study. Section 2.4 discusses the learning process as it occurs within a CILT-MAL system. Finally, Section 2.5 places the learning agents used in this CILT-MAL system, into Langley's framework for machine learning and draws a comparison between the two frameworks.

## 2.1 Co-operative inductive learner teams

In 1912, Russell [in Sestito *et al* 1994] defined the induction principle as:

*"When a thing of a certain sort A has been found to be associated with a thing of a certain other sort B, and has never been found dissociated from a thing of the sort B, the greater the number of cases in which A and B have been associated, the greater is the probability that they will be associated in a fresh case in which one of them is known to be present".*

Later, in 1988, Honavar and Uhr [in Sestito *et al* 1994] defined the induction process as:

*"A process by which a system develops an understanding of principles or theories that are useful in dealing with the environment by generalization and specialization from the specific examples or instances presented to the system. This includes the process of experimentation and discovery; that is setting the hypothesis and then accumulating evidence to confirm or deny their validity".*

Inductive learning is *the construction of a description of a problem domain, based on the observations made of that domain, by a learner*. The automation of this construction process is referred to as inductive machine learning [Holsheimer *et al* 1994]. Data mining is a special case of inductive machine learning, where the problem domain is being observed through a data repository. This data repository describes the problem domain in the following way:

Let $A = \{A_1, A_2, \ldots\ldots, A_n\}$ be a set of attributes with domains $Dom_1, Dom_2, \ldots\ldots, Dom_n$. A data repository $S$ is a table over $A$, with an example or fact being a tuple in the repository $S$. Consider the example in Table 1 :

**Table 1 Vehicle descriptions**

| Make | # of wheels | Wings | Class |
|---|---|---|---|
| Toyota | 4 | No | Car |
| Boeing | 3 | Yes | Airplane |
| Honda | 2 | No | Motorbike |
| Sailboat | 0 | No | Boat |

Each attribute in the table describes a type of vehicle using all the properties of the tuple.

$A_1$ is the make,

$A_2$ denotes the number of wheels,

$A_3$ specifies whether it has wings or not, and

$A_4$ the class.

Hence

*Dom*$_1$ = *{Toyota,Boeing,Honda,Sailboat},*

*Dom*$_2$ = *{4,3,2,0},*

*Dom*$_3$ = *{no,yes,no,no}, and*

*Dom*$_4$ = *{car,boat,motorbike,airplane}.*

The data repository contains three attributes that denote the class of the tuple, i.e. $A_1$, $A_2$, and $A_3$. These are called predicting attributes. $A_4$ is the predicted attribute. A training set, i.e. the set of training examples, is a subset of the repository S presented to the learners through which the problem domain is being observed. A combination of values for the predicting attributes defines a class. A class $C_i$ is a subset of the database $S$, consisting of all objects that satisfy the conditions, ***cond***$_i$ , that define the class description $D_i$ [Holsheimer *et al* 1994]. Therefore, a class $C_i$ may be expressed as:

$C_i = \{o\ 0\ S/cond_i(o)\}$.

Inductive machine learning can be defined as *the process where a hypothesis, i.e. a rule or a set of rules, is formed by observing the environment through a data set formatted as (predicting attributes, predicted attribute). The rules describe the input features of a problem domain that will predict a certain output and can be interpreted as classifiers or trends*. All the learners', which participated in this study, learning technique is based on the induction principle, therefore they are referred to as inductive learners.

During this research, the participating learners did not learn in isolation. They co-operated with one another during the learning process. Over the past 12 years there has been a keen interest in applying principles of co-operation into human learning environments (Slavin 1983). Johnson (1994) defines human co-operative learning as *"the instructional use of small teams so that students work together to maximise their own and one another's learning"*. A wide variety of human co-operative learning techniques have been developed and evaluated. These techniques have two primary components namely, a co-operative incentive structure and a co-operative task structure. Slavin (1983) defines a co-operative incentive structure *"as a structure in which two or more individuals are rewarded based on their performance as a group. Co-operative incentive structures usually involve co-operative task structures, which are situations in which two or more individuals are allowed, encouraged or required to work together to complete a task"*. For example, when three people travelling in a car push the car out of a ditch, all of them benefit from each other's efforts. Either all of them will be rewarded, by being able to continue their journey, or none of them will be, depending whether they succeed or not. Hence, when these techniques are applied to the world of machine learning, co-operative inductive learning refers to the active co-operation of two or more inductive learners in a learner team, to acquire new knowledge, to maximise their individual and/or combined results. The level of co-operation among the learners in a learner team may differ, for the purpose of this study an approach of full co-operation was used.

## 2.2 Multi-agent learning system

In Section 1.1 it has been stated that learning environments can be viewed from different perspectives and that this study will follow Viktor's (1999) approach and model co-operative inductive learner teams as multi-agent learning systems. Russell *et al* (1995) defines an agent as *"anything that can be viewed as perceiving its environment through sensors and acting upon that environment through effectors"*. An agent that can act rationally at any given time is perceived as being successful. According to Russell *et al* (1995), four factors influence an agent's ability to act rational: A performance measure that determines how successful the agent is; A percept sequence, which includes everything the agent has perceived thus far; The agents background knowledge of the environment; And lastly, the actions that the agent is capable of performing. A rational agent can therefore be defined as *an agent that for each possible percept sequence, will do whatever action is expected to maximise its performance measure, based on the evidence provided by the percept sequence as well as whatever built-in background knowledge of the environment the agent has* (Russell *et al*, 1995;Knapik *et al*, 1998; Jennings *et al*, 1998). A learning agent is a rational agent with the added ability to improve itself by interacting with its environment and by observing its own decision making process.

Weiss (1998) defines a multi-agent learning system *"as a system that consists of a number of learning agents residing in a communal environment that interact with one another as well as the environment"*. Hence, a co-operative inductive learner team can be modelled as a multi-agent learning system, with co-operative inductive learners participating as learning agents that interact with one another by means of full co-operation, as discussed in Section 2.3.

Hand [in Berthold *et al* 1999] argues that intelligent data analysis has two significant parts, one being the learning environment concerned with finding a model for the data, the other concerned with the actual algorithms, i.e. the computational facilitators that enable the environment to analyse the data. The following two sections, Section 2.3 and Section 2.4, address the first significant part, namely defining the learning environment. Section 2.5 addresses the second significant part, the algorithms, by placing the different learning agents into Langley's framework for machine learning. Langley's framework defines machine learners in terms of four components, each describing the key aspects that influence a learner's learning process.

## 2.3 Modelling a co-operative inductive learner team as a multi-agent learning system

Recall from Section 2.2 that, a multi-agent learning system consists of a number of learning agents that reside in a learning environment. The learning agents interact with one another, as well as the environment [Weiss 1998]. Within a co-operative inductive learner team the learners interact with the environment by accepting training material, such as training examples, from the environment and feed discovered knowledge, such as rules induced from the training material, back to the environment. This section gives an overview of the CILT-MAL system.

In the CILT-MAL system, the learners interact with one another by means of full co-operation. Viktor (1999) describes full co-operation as *"the method whereby all the participating learners, i.e. members of a learner team, receive the same study material and share the knowledge acquired with one another in such a way that the performance of all the learners in the team improves"*. The inductive learners form co-operative inductive learner teams within which they co-operate fully with one another. This is achieved as follows: all the learners that participate in a learner team are presented with a training set, as defined in Section 2.1. These sets are identical. Each learner executes an individual inductive learning episode, also known as a single-agent episode, during which no communication with other learners occurs, as described in Section 2.4.1. This is followed by the two evaluation episodes of the individual learning phase. Co-operative learning, as described in Section 2.4.2, follows during which the learners, within each co-operative inductive learner team, share the knowledge acquired during their individual learning phase with each other. The learners then re-execute their individual learning phases with the shared knowledge incorporated into each learners training set, which can again be followed by a co-operative learning episode. This iterative process continues until all the learners have mastered the material at hand and each team has reached consensus that the learning episode has successfully been completed. Finally, as a performance measurement, the knowledge acquired by each learner is validated against a set of unseen training examples called the validation set, as described in Section 2.4.3. The validated knowledge generated by the different teams is then fused together to represent the knowledge acquired by the learning environment as a whole.

Learners sharing their acquired knowledge, i.e. induced rules, with one another accomplish full co-operation. Hence, the learners are able to improve their behaviour through the experience gained by learning from the training material as well as the knowledge gained by co-operating with other learners. An inductive learner residing within a co-operative inductive learner team can be modelled as a learning agent residing in a multi-agent learning system, and referred to as a co-operative inductive learning agent, as shown in Figure 2.

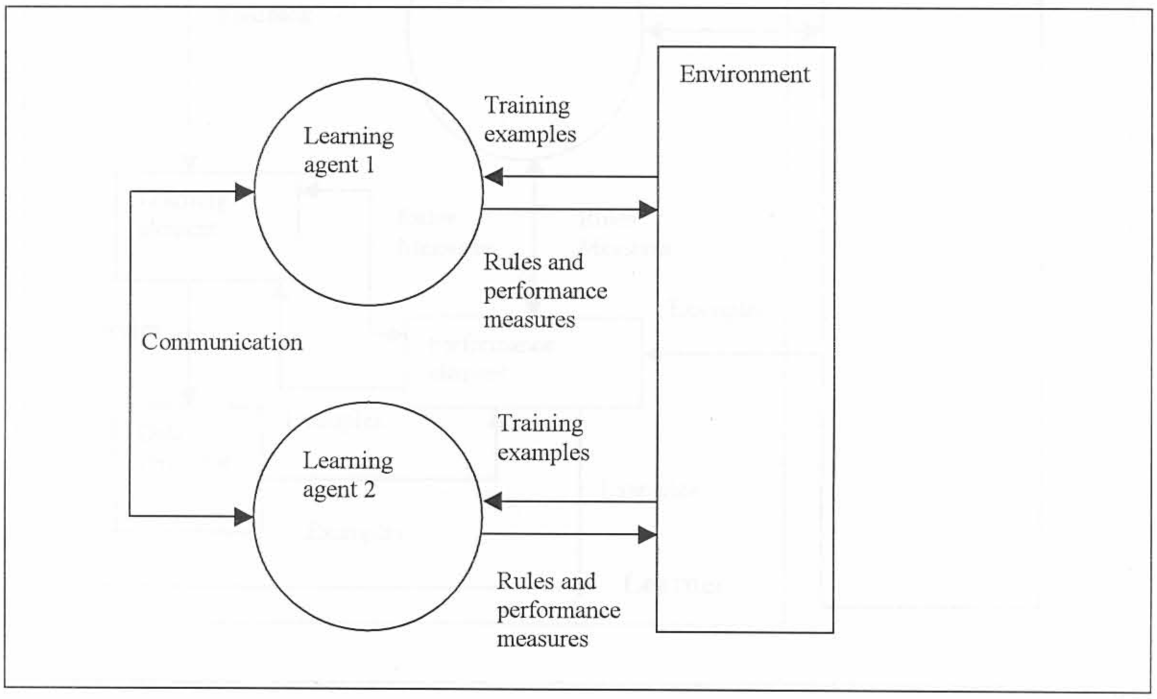

**Figure 2 Co-operative learner team modelled as a multi-agent learning system**

Russell *et al* (1995) defined a general model for learning agents. The model divides a learning agent into four conceptual components namely, the learning element, performance element, critic and problem generator. Viktor (1999) adapted Russell *et al*'s general model and defined the inductive machine learning architecture for modelling co-operative inductive learning agents. The inductive machine learning architecture divides each learning agent into five conceptual components namely, the learning element, performance element, critic, data generator and knowledge base. Figure 3 shows the modelling of a co-operative inductive learning agent using the inductive machine learning architecture.

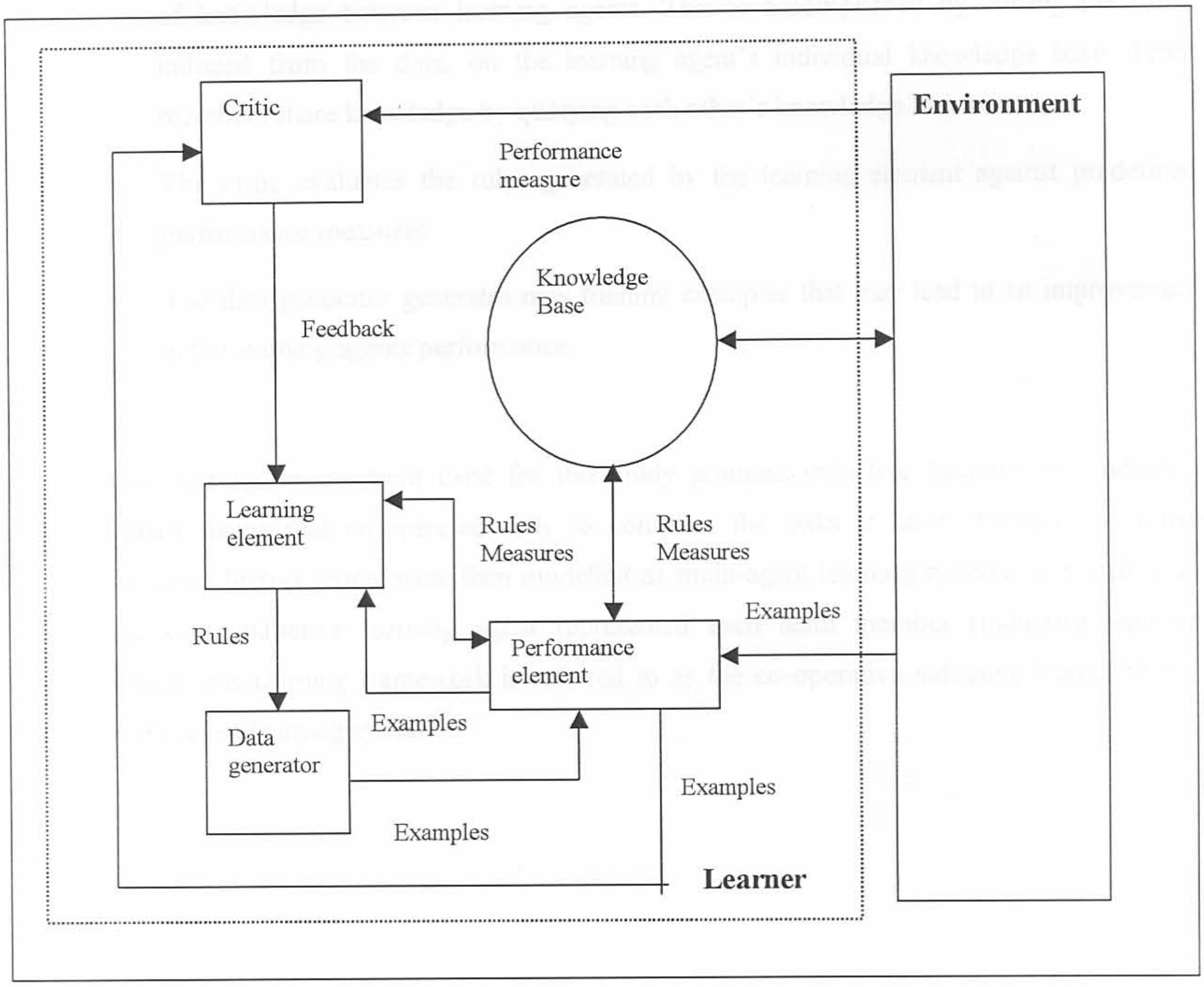

**Figure 3 Model of a co-operative inductive learning agent using the inductive machine learning architecture [Viktor 1999]**

The role of the five components are described as follows [Viktor 1999]:

- The learning element contains the inductive machine learning process that produces sets of rules by observing the environment through a data repository. These rule sets describe the problem domain.
- The performance element controls, monitors and guides the progress of the learning element by accepting training material in the form of training examples from the environment and presenting them to the learning element or critic. It also controls the communication of rules and measures to other components.

- The knowledge base is used for inter learner communication by facilitating the transfer of knowledge between learning agents. This is accomplished by storing the rules, induced from the data, on the learning agent's individual knowledge base. Team members share knowledge by querying each other's knowledge bases.
- The critic evaluates the rules generated by the learning element against predefined performance measures.
- The data generator generates new training examples that may lead to an improvement in the learning agents performance.

The learning environment used for this study grouped inductive learners into inductive learner teams that co-operated fully to complete the tasks at hand. These co-operative inductive learner teams were then modelled as multi-agent learning systems, in which a co-operative inductive learning agent represented each team member (inductive learner). Hence, this learning framework is referred to as the co-operative inductive learner team - multi-agent learning system.

## 2.4 Learning in the CILT-MAL system

Figure 4 graphically depicts the learning process as it occurs within the CILT-MAL system.

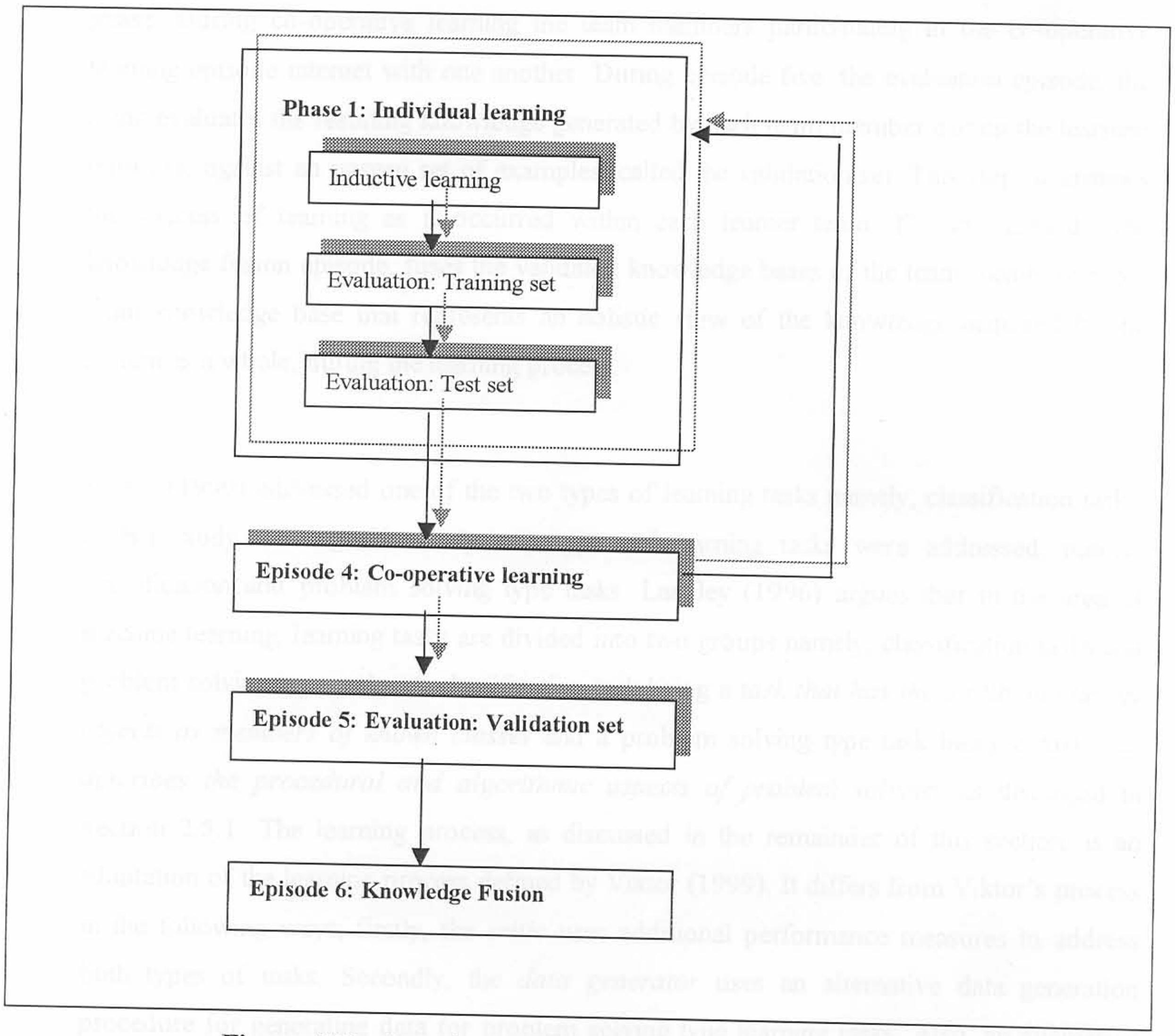
Phase 1: Individual learning
Inductive learning
Evaluation: Training set
Evaluation: Test set
Episode 4: Co-operative learning
Episode 5: Evaluation: Validation set
Episode 6: Knowledge Fusion

**Figure 4 The learning process of the CILT-MAL system**

This learning process is controlled by the *performance element*, which controls and monitors the progress of the *learning element*. Learning occurs in two phases. The first, being individual learning, also known as single-agent learning, during which the learning agents learn without interacting with team members. The individual learning phase consists of three episodes namely, episode one, the inductive learning episode. Followed by episode two, an evaluation episode during which evaluation occurs against the training set. The individual learning phase concludes with episode three, during which evaluation occurs against the test set.

Co-operative learning, also known as multi-agent learning, follows the individual learning phase. During co-operative learning the team members participating in the co-operative learning episode interact with one another. During episode five, the evaluation episode, the *critic* evaluates the resulting knowledge generated by each team member during the learning episodes, against an unseen set of examples, called the validation set. This step determines the success of learning as it occurred within each learner team. The last episode, the knowledge fusion episode, fuses the validated knowledge bases of the team members into a team knowledge base that represents an holistic view of the knowledge acquired by the system as a whole, during the learning process.

Viktor (1999) addressed one of the two types of learning tasks namely, classification tasks, in her study. During this study two types of learning tasks were addressed, namely classification and problem solving type tasks. Langley (1996) argues that in the area of machine learning, learning tasks are divided into two groups namely, classification tasks and problem solving type tasks. A classification task being a *task that has the ability to classify objects as members of known classes* and a problem solving type task being a *task that describes the procedural and algorithmic aspects of problem solving,* as discussed in Section 2.5.1. The learning process, as discussed in the remainder of this section, is an adaptation of the learning process defined by Viktor (1999). It differs from Viktor's process in the following ways; firstly, the *critic* uses additional performance measures to address both types of tasks. Secondly, the *data generator* uses an alternative data generation procedure for generating data for problem solving type learning tasks. Also, an alternative rule pruning technique for problem solving type learning tasks was added and lastly an additional episode, knowledge fusion, was used at the end of the learning process.

### 2.4.1 Individual learning phase

During single-agent learning all the learners receive the same study material, i.e. an identical training set of examples. This training set should contain 70% of the instances, randomly chosen form the data repository, describing the problem domain [Theron 1993]. Following Theron (1993), each experiment's individual inductive learning episode is repeated at least 10 times, with randomised training sets. Each of the individual *learning elements* analyse their own training set to induce the best, most accurate set of rules describing the problem domain, without any interaction with the other learners. The *critic* then evaluates each rule set against the training set to measure the learners' performance against known cases.

The *critic* uses the following performance measures, determined by the type of learning task, to evaluate the performance of the learners.

- For a classification task the overall rule set accuracy is used as the performance measure. The overall accuracy of a rule set, covering class descriptions $D_1, D_2, D_3, \ldots D_i$, describing classes $C_1, C_2, C_3, \ldots C_i$, is the number of instances in the set that are covered correctly, expressed as a percentage of the total number of instances in the set. Section 2.5.4.1 contains a detailed description of these measures,
- For problem-solving type tasks the strength, interestingness and consistency of each individual trend are used as the performance measures. The strength of a trend is measured by the percentage positive coverage for that trend. The interestingness of a trend is measured by the difference in the percentage coverage of a trend over different classes, the bigger the difference the more interesting the trend. Lastly, the consistency of a trend is determined in terms of how consistent the strength of the trend is when evaluated against different data sets. A detailed description of these measures can be found in Section 5.4.

Next, the *critic* evaluates each rule set against the unseen test cases. A test set, consisting of 15% of the instances that describes the problem domain, are randomly chosen form the data repository [Theron 1993]. The performance element presents this test set to the *critic*. The *critic* evaluates each rule against the test set, using the above performance measures. The resulting rule set of each learner as well as the performance measures when evaluated against the test set, are stored in each individual learner's *knowledge base*. Next, the learners engage in co-operative learning.

### 2.4.2 Co-operative learning episode

Once all the learners have executed their individual learning phases, those participating in co-operative inductive learner teams, engage in the co-operative learning episode. The co-operative learning episode of a multi-agent learning system involves the improvement of the performance measures of individual rule sets through co-operation. This is accomplished by replacing a low quality rule in a learner's knowledge base with a related high quality rule, obtained from a team member's knowledge base. This co-operative learning episode is executed as follows.

Firstly, each learner evaluates its own knowledge base to identify all the low quality rules, i.e. rules with performance measures below some predetermined threshold value. The learner then queries the knowledge bases of all the other learners in the team for high quality rules, i.e. rules with performance measures above or equal to some predetermined threshold value, related to the low quality rules identified previously. Viktor (1999) categorises related rules into three categories namely, rules that overlap each other, subsume each other or correspond to each other. That is, a low quality rule can be identified as a misconception or being in conflict with a high quality rule and therefore be removed from the learner's knowledge base. These relating high quality rules are then placed on a NewRule list. Hence, all the low quality rules are removed from the knowledge base.

Secondly, the *data generator* subsequently generates new training sets for each rule on the NewRule list in the following way.

- For a classification task, following Viktor (1999), the data generator generates a strongly biased training set for each high quality rule on the NewRule list. Each of the new training sets contain the same class distribution as the original training set, however, the attribute values of those attributes that form part of the high quality rule

i 16116902
b15483113

are biased towards the new rule. Those attributes that do not form part of the rule's attribute tests are distributed in exactly the same way they were in the original training set. This ensures a new training set strongly biased towards the high quality rules without changing the characteristics of the remainder of the training set.

- However, if the learning task is a problem solving type task one single training set is generated for all the high quality rules on the NewRule list. The approach used for the classification task generates a strong biased data set for each high quality rule on the NewRule list. This leads to the production of rules that do not necessarily reflect the knowledge as contained in the original training set [Viktor 1999]. For a problem solving type task the data is generated in the following way. For any rule $R_1$ that covers $[x_1, x_i]$ instances of the training set (where $x_1$ is the number training instances covering class $C_1$ and $x_i$ the number of training instances describing one or more of the other classes $C_i$ as contained in the data set) the data generator produces a new training set that contains $x_1$ new occurrences of the class $C_1$ and $x_k$ new occurrences of each of the other classes $C_i$ , where rule $R_1$ covers $x_k$ examples of class $C_i$ . Following this approach, the coverage of the high quality rules double and the remaining instances, which are not covered by the new high quality rules, are left as is. Therefore, instead of generating one strongly biased data set for each high quality rule, one single data set is generated with twice the number of instances covering the high quality rules on the NewRule list. The performance of a problem solving type task is measured by the strength, interestingness and consistency of single trends, which is based on the trends percentage coverage, rather than its prediction accuracy. Therefore, this approach to data generation is better suited for this type of task.

After the data generation process, the learner returns to the individual learning phase to generate new sets of rules by learning from the newly created training sets. This leads to newly generated high quality rules that are then added to the learner's individual knowledge base. The system follows an incremental learning approach where the knowledge bases are improved by adding the high quality results of the various iterations. The co-operative learning episode continues until the solution to the learning problem cannot be improved or a predetermined period of time has elapsed.

Finally, a rule-pruning algorithm is used to simplify the individual knowledge bases. Once again the type of learning task determines the pruning algorithm that will be used.

- For classification type tasks the study follows Viktor (1999) and uses the reduced error-pruning scheme (REP), as introduced by Quinlan (1994).
- For problem solving type tasks performance is measured in terms of each individual trend's qualitative measures namely, strength, interestingness and consistency and not in terms of the overall rule set accuracy as is the case for classification type tasks. This makes the REP pruning algorithm not applicable. When simplifying the knowledge bases of a problem solving type task, duplicate trends should be removed. Any further simplification, for example the combining of trends, is dependent on user needs and not on quantitative measures.

Next, the resulting rule set of each learner as well as the performance measures when evaluated against the validation set, are stored in each individual learner's *knowledge base*.

### 2.4.3 Evaluation and knowledge fusion episodes

The fifth episode involves the validation of the resulting sets of high quality rules, generated during the co-operative learning episode. These rule sets are evaluated against the previously unseen set of validation, i.e. the remaining 15% of the instances, randomly chosen form the data repository. The performance measures, when evaluated against the validation set, are stored in each individual learner's *knowledge base*.

During the final stage, the high quality rule knowledge bases of each team member are fused together into a team knowledge base. The co-operative incentive structure, as introduced in the beginning of this chapter, of this co-operative learning technique is to acquire new knowledge, for the maximisation of individual and/or combined results. Therefore, each team's knowledge base is pruned and tested against a validation set to determine the team's success as a group. Finally the knowledge bases of the participating teams are fused together into one holistic knowledge base representing the knowledge acquired by the system as a whole.

Recall from Section 2.2 that an intelligent data analysis has two significant parts, one being the learning environment concerned with finding a model for the data, as discussed above. The other concerned with the actual algorithms, i.e. the computational facilitators that enable the environment to analyse the data, as discussed next.

## 2.5 Langley's framework for machine learning

The multi-agent learning systems constructed during this study used a variety of different learning agents. Each learning agent had a learning element component that contained the inductive machine learning process that produced sets of rules that described the problem domain. For the purpose of interpreting and comparing the individual rule sets produced by the different agents, their learning elements, i.e. algorithms, were modelled according to Langley's (1996) framework for machine learning.

Langley (1996) defines machine learning as *"the **improvement** of **performance** in some **environment** through the acquisition of **knowledge** resulting from **experience** in that environment"*. Langley describes machine learning as an ongoing process during which a machine learner's execution of a specific task, within a specified problem domain will improve as long as the learner is presented with sufficient data that describes the problem. Whereas:

- **improvement** involves a desirable change in a performance measure;
- **performance** suggests some quantitative measure of behaviour on a task, such as efficiency, accuracy, or understanding;
- the **environment** is an external setting with regularity;
- **knowledge** as depicted through a kind of internal data structure;
- **experience** requires some processing, depicted by the learning mechanism itself.

According to Langley (1996), learning does not happen in isolation. Rather, it is the intricate interaction between all these components as illustrated in Figure 5.

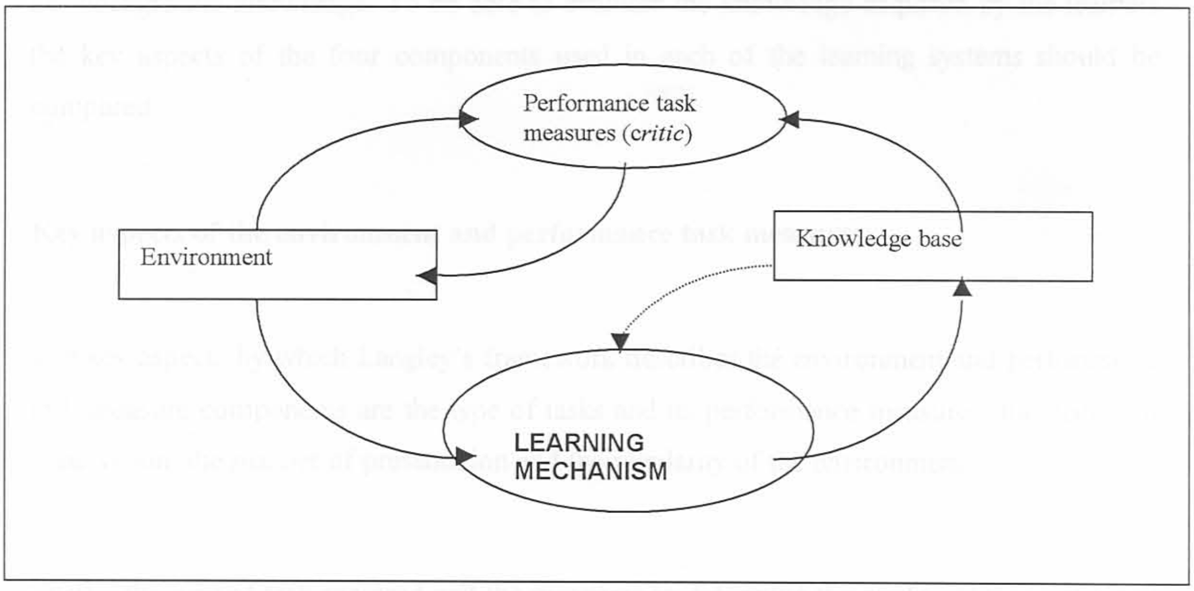

**Figure 5 Langley's machine learning framework**

The figure shows that learning always takes place in an environment, about which a learner attempts to learn something. On the other hand, a learner is always linked to a knowledge base from which it can extract previously acquired knowledge or experience (the input to learning), and into which it can store new acquired knowledge (the output of learning). The success of the learning mechanism is measured by the improvement of the task's performance. The performance task measurer uses the knowledge base to control its interactions with the environment.

Langley uses these four components namely: the environment, performance task measures, learning mechanism and knowledge base to describe a learning system, i.e. a learner or a learning agent. Each one of these components incorporates key aspects that influence the learning process. Langley's machine learning framework relates to the inductive machine learning architecture, as discussed in Section 2.3, in the following way. The performance task measures correspond to the criteria by which the *critic,* of the inductive machine learning architecture, measures the performance of a *learning element.* The environment corresponds to the *environment* residing outside the learning system. The learning mechanism corresponds to the combined role of the *learning* and *performance elements* of the inductive machine learning architecture. The knowledge bases contain the acquired knowledge for both of the architectures but differ in the following way, as for Langley's framework the knowledge base also contains previously acquired knowledge or experience i.e. background knowledge. To be able to evaluate the knowledge acquired by the learners the key aspects of the four components used in each of the learning systems should be compared.

## 2.5.1 Key aspects of the environment and performance task measures

The key aspects by which Langley's framework describes the environment and performance task measure components are the type of tasks and its performance measures, the degree of supervision, the manner of presentation and the regularity of the environment.

Firstly, the type of task required and the measures to determine the quality of the performed task are used to describe the learning environment. Learning tasks can be divided into two groups namely, classification tasks and problem solving type tasks i.e. those that use acquired knowledge for classification purposes and those that use the learned knowledge for some form of problem solving [Langley 1996]. A classification task, which is a *task that has the ability to classify objects as members of known classes*, for example, when analysing a soybean data set, containing descriptions of instances carrying different soybean diseases, the aim of the task is to classify the soybeans according to their diseases. A problem solving type task, which is a *task that describes the procedural and algorithmic aspects of problem solving*, for example, when given a set of constraints describing a system, should be able to optimise the system's parameters. The goal of the inductive learners is to increase the accuracy of the performance system, regardless the type of task. For example, the performance measure of a classification task is the accuracy by which it can predict the class of unseen cases.

Secondly, the learning environment is described by means of the degree of supervision. The degree of supervision determines whether supervised or unsupervised learning is occurring. During supervised learning, the learning mechanism (learning element) is presented with the training material that includes both the predicting attributes, as well as a preferred solution i.e. the predicted attribute, as defined in Section 2.1. The learning element should find a path from the predicting attributes to the preferred solution. However, during unsupervised learning the learning element is presented with training material that includes only the predicting attributes. Hence, the learning element should find the best possible solution as well as the different paths to get from the training material to the solution.

Thirdly, the manner of presentation describes the way in which the training examples are presented to the learning mechanism i.e. how the *performance element* presents the data to the *learning element*. All the examples can be presented simultaneously which is referred to as offline learning, or one at a time, which is referred to as online learning.

Lastly, the regularity of the environment refers to the complexity of the target knowledge that should be acquired based on the number of irrelevant attributes, the amount of noise in the environment and the consistency of the environment over time.

### 2.5.2 Key aspects of the knowledge base

The knowledge base within Langley's framework is responsible for the representation of both experience (input to learning) and acquired knowledge (output of learning), which differs from that of the inductive machine learning architecture. The knowledge base of the inductive machine learning architecture stores the acquired knowledge only, the rule sets and their performance measures.

The first aspect of the knowledge base component is the way in which training; test and validation sets are formatted. For example, it could be in binary format, as a set of nominal attributes, numeric attributes, or as a set of relational literals. The second aspect is the way in which the acquired knowledge produced by the learning mechanism is presented. This representation involves the formulation of a description that can distinguish between instances belonging to a specific class and those that don't. These descriptions are formulated using a concept description language. The interface between the concept description and the actual instances is referred to as an interpreter or matcher. The role of the interpreter is to use the concept descriptions, described in terms of a concept description language, to classify instances, represented in a specific format.

Interpreters are categorised under one of three approaches, namely, the logical approach, threshold approach or competitive framework. The logical approach uses an "all or none" match to determine whether an instance belongs to a certain class. The threshold approach uses a partial matching process to classify instances, and the competitive framework uses a best-match approach. Langley emphasises that the interpretative approach far outweighs the importance of representation, since different interpreters can yield different meanings for the same representation.

The third aspect is that of organisational structures. In the field of machine learning one of three classes of organisational structures is used to represent the knowledge acquired about static objects. They are decision lists, inference networks or concept hierarchies.

### 2.5.3 Key aspects of a learning mechanism

Langley (1996) argues that learning is a search through a problem domain. He therefore describes the learning element in terms of the incrementality of the learning process, its search technique and search bias. The incrementality of the learning process refers to the manner in which training examples are processed, all at once in a non-incremental manner, or one at a time, in an incremental manner.

The way in which learning agents apply search operators to a problem domain is referred to as the search technique. These search techniques are described firstly, in the way they start the search, secondly, how the search is organised i.e. the search bias and how alternative states are evaluated and finally, when to terminate the search.

Induction methods typically start their search through the problem domain from either the most general or the most specific state, although some may start from a randomly selected state. When starting from the most general state, the concept description describing the problem domain will cover all of the instances in the training set i.e. the rule set will be empty. When starting from the most specific state, the concept description describing the problem domain will be so specific that it will cover the smallest number of instances in the training set. The search can be organised to explore the search space using exhaustive methods or heuristic methods. Exhaustive searches are inefficient [Russell *et al* 1995] since every possible path is explored. Heuristic methods, also known as informed search methods, are therefore preferred. These methods use an evaluation function that determines the desirability of a specific search path. Heuristic methods sacrifice optimality but in return gain in tractability. Search procedures terminate when no further progress is made or when the method finds a concept description that is entirely consistent with the training data.

A learning system must direct or limit its search through the space of possible concept descriptions. These directions or restrictions are often referred to as the search bias of the system. The bias of the learning element can be a representational bias, which restricts the solution space by limiting the concept description language. Or, the bias may be a search bias, which considers all possible concept descriptions for a given problem domain, but chooses to examine some descriptions earlier than others in the search process. Sometimes these preferences are encoded into the evaluation metric of the learning element, or else they form part of the structure of the search algorithm itself. Another source of bias can be that of background knowledge available to the learning system, which can strongly guide the course of learning.

A co-operative inductive learning agent, as described in Section 2.2, can be modelled using the inductive machine learning architecture, as employed by Viktor (1999). To further define the learning process, with an emphasis on the learning element the learning agents are placed into Langley's machine learning framework. The next four sections describe the four learning agents used for this study, CN2, C4.5, BRAINNE and the human learner, using Langley's machine learning framework.

## 2.5.4 CN2 – A rule induction learning agent

CN2 is a rule induction program developed by Clark *et al* (1989) designed for the efficient induction of simple, comprehensible rules in problem domains where problems of poor description language and/or noise may be present.

### 2.5.4.1 Environment and performance task measures

CN2 typically performs classification tasks (as defined in Section 2.5.1). The aim of CN2 is to induce simple concept descriptions that accurately classify novel instances. The input to the system is presented as a training set of examples, where each instance is described as a set of numerical attributes, including an attribute that specifies its class. This indicates that offline supervised learning, as defined in Section 2.5.2, is occurring. The program has an evaluation function that evaluates the accuracy of the individual rules as well as the overall rule set, as follows:

- Individual rule accuracy

  As an example the inductive learner generated the following rule during the execution of the problem solving type task, as discussed in Chapter 5.

```
IF    Highest_qualification = Degree
  AND Gender = F
  AND Discipline_specialisation = AMS
THEN  Org_type = Technikon   [11 10 12]
```

  The CN2 evaluation function evaluated the rule, on an unseen validation set consisting of 212 instances, as follows:

```
EVAL> individual
                  FIRED?
ACTUAL CLASS Yes      No      Accuracy
 Technikon     3       10        23.1 %
Not Techni     2      197        99.0 %
Overall accuracy: 94.3 %
```

Where the accuracy of the individual rule is calculated as the sum of the number of correct positive covered instances, i.e. 3, and the correct negative covered instances, i.e. 197, as a percentage of the total number of instances read i.e. 212.

(3 + 197)/212 * 100 = 94.3%

- Overall rule set accuracy

As an example the CN2 evaluation function evaluates the overall rule set as follows:

```
                         PREDICTED
ACTUAL        Private Technik Univers Accuracy
  Private      164       0       0     100.0 %
Techniko        10       0       3       0.0 %
Universi        33       0       2       5.7 %
Overall accuracy:  78.3 %
```

Where the accuracy of the rule set is calculated as the sum of all the correct positive covered instances, i.e. 100, 0, 2, as a percentage of the total number of instances, i.e. 212.

(100 + 0 + 2)/212 * 100 = 78.3%

### 2.5.4.2 The knowledge base and performance element

CN2 organises its output as a list of if-then rules, also known as a decision list. The concept description language describes each rule induced by CN2 as follows:

–'if <complex> then predict <class>',

–<complex> is a conjunct of <attribute test>,

–<attribute test> is a test on a single attribute.

For example, the CN2 learner generated the following rule during the analysis of a business classification task, as described in Chapter 4.

```
IF     Process_Capability < 5.50
   AND SIC_Code_Old > 3560.00
THEN   company_sector = Product  [79 0]
```

Where `IF Process_Capability < 5.50`
`AND SIC_Code_Old > 3560.00` is the complex,

`Process_Capability < 5.50`
`SIC_Code_Old > 3560.00` are attribute tests and

`company_sector = Product` is the class.

The interpreter applied by CN2 falls into the logical class approach, carrying out an "all or none" matching process. Each rule is tried until one is found whose conditions are all satisfied by the instance being classified. If no induced rule is satisfied, the default class, as determined by the learning mechanism as the class that most instances in the data set belongs to, is assigned to the instance.

### 2.5.4.3 The learning mechanism

The CN2 learning algorithm searches the problem space by means of a pruned general-to-specific search, starting at the most general concept description, as described in Section 2.5.3. At each stage of the search the algorithm retains a size-limited set of best complexes found thus far. The star size parameter (default 5) defines the limit of complexes that are retained during execution. This set is further specialised by applying a beam search to the space of complexes. The complexes are evaluated so that the best, reliable complex may be found. This process iterates until no more good, reliable complexes, as defined next, are found.

The search is directed by two heuristic decisions. Firstly, the quality of the complexes found is assessed. This is done by means of an information-theoretic entropy measure. This measure prefers complexes covering a large number of examples of a single class and few examples of other classes, such as the complex in the above-mentioned example that covers 79 instances of the "Product" class and none of the other classes. Secondly, the significance of the complex is measured. This measure attempts to establish whether a complex reflects a genuine correlation between attribute values and classes, or if the correlation just occurred by chance. The user can manipulate this relationship by setting the significance threshold (default 0). The two heuristic decisions based on entropy and significance determines whether the complexes found during a search are both of good quality and with high correlation.

## 2.5.5 C4.5 – A decision tree learning agent

C4.5 is a decision tree generation program developed by Quinlan (1994). C4.5 is an industrial strength version of ID3 developed by Quinlan (1994) to investigate the effect of noise on learning. Recently Quinlan released C5.0 an updated version of C4.5 (www.rulequest.com).

### 2.5.5.1 Environment and performance task measures

The program takes a set of examples as input and generates a decision tree that classifies the training set as output. The program has an evaluation function that evaluates the accuracy of the generated classifier on pre-classified test examples. Since large decision trees are too complex to be understood, C4.5 has a conversion function that re-expresses the generated decision tree as a decision list.

C4.5 is mostly used as a classifier that aims to build a compact decision tree, which is consistent with the training set and reveals the structure of the domain in such a way that it has predictive power. The input set of examples includes an attribute that specifies the class of the instances, which indicates that offline supervised learning is occurring.

### 2.5.5.2 The knowledge base and performance element

All the instances in the training set are described as a set of numerical attributes. An attribute may have either a discrete or a numeric value. The class to which the instance belongs must be established beforehand and is denoted as the last attribute value. C4.5 organises the output as a decision tree, also known as a concept hierarchy, consisting of decision nodes, branches and leaves. The concept description language describes each node or leaf induced by C4.5 in the following format:

– a decision node: specifies a test on a single attribute, with one branch for each possible outcome:

‘<attribute test>

<attribute test> / leaf’

– leaf: specifies a test on a single attribute and indicates a class.

‘<attribute test> : predicted class’

– <attribute test> is a test on a single attribute.

For example, the following decision tree was generated by C4.5 during the problem solving type task described in Chapter 5.

```
Gender = M: Engineering (577.0/253.8)
Gender = F:
|   Highest_qual = Dipl: APS (53.0/33.9)
|   Highest_qual = Postgrad: APS (49.0/34.7)
|   Highest_qual = Degree:
|   |   Race = A: ABS (65.0/37.3)
|   |   Race = D: APS (9.0/6.4)
```

Where `<Gender = M: Engineering>` is a leaf,

```
<Gender = M: Engineering
 Gender = F:                  >       is a node and
<Highest_qual = Degree>               is an attribute test.
```

The decision tree interpreter applied by C4.5 decides to which class an unseen instance belongs, given the instance’s attributes. The goals set for the interpreter are to permit the use of imprecise information about attribute values and to estimate the degree of certainty associated with the predicted, most likely class alternatives. The C4.5 decision tree interpreter falls into the logical class of approaches, carrying out an "all or none" matching process. However, when the soft threshold option is invoked during the construction of the decision tree the interpreter falls into the threshold class of approaches carrying out a partial matching process [Quinlan 1994]. An instance is classified by testing its attribute values against the decision nodes, starting at the root and working its way up the tree until it satisfies the condition of a leaf, at which point the predicted class of that leaf will be assigned to the instance. By taking into consideration the predicted number of errors at a leaf more than one leaf can be satisfied by the attribute values of an instance. Therefore, the interpreter calculates an estimated certainty factor for each possible classification to indicate the most likely class. The interpreter handles imprecise values in a similar way to the learning algorithm's way of handling unknown values, which is described in the next section. If the soft threshold option is invoked during tree construction a weighting method is used to soften absolute thresholds. For each continuous attribute an interpolation range is calculated, so that if the attribute value lies within the range both outcomes will be explored with a calculated weight assigned to each [Quinlan 1994].

C4.5 organises the output as an ordered decision list. By invoking the conversion function, called C4.5rules. Each path, from the root of a decision tree to a leaf, will be converted into one initial rule. The concept description language describes each rule in the following format 'if <complex> then predict <class>', where <complex> is a conjunct of single attribute tests. The decision list interpreter applied by C4.5 falls into the logical class approach, carrying out an "all or none" matching process. The rules are tried in the specified top-down order until one is found whose conditions are all satisfied by the instance being classified. The interpreter then calculates the estimated certainty factor for the suggested class. The following rules were extracted form the decision tree listed previously in this section.

The final rules from decision tree:

```
Rule 5:
     Highest_qual = Postgrad
     Race = D
->   class APS   [41.7%]
```

```
Rule 3:
    Highest_qual = Dipl
    Gender = F
->  class APS  [36.0%]

Rule 2:
    Race = D
    Gender = F
->  class APS  [34.5%]

Rule 4:
    Race = A
    Gender = F
->  class ABS  [32.6%]

Rule 7:
    Highest_qual = Dipl
    Gender = M
->  class Engineering  [61.1%]


Rule 6:
    Highest_qual = Degree
    Gender = M
->  class Engineering  [61.1%]

Default class: Engineering
```

### 2.5.5.3 The learning mechanism

The C4.5 learning algorithm searches the problem space by means of a greedy method that directs the search with an evaluation function. Greedy methods are the simplest form of best-first search strategies. A best-first strategy's decision on which node to expand during a search, is based on a value produced by an evaluation function. The node obtaining the best evaluation will be expanded first. When the evaluation function evaluates the estimated cost of a path from a node to its goal, the search is conducted by means of a greedy method. That is, the node whose state is judged to be closest to the goal state is always expanded first [Russell *et al* 1995].

C4.5 organises the search from the root node downward, in a divisive manner. The algorithm follows a divide and conquer-approach, in an attempt to partition the training set. Firstly the algorithm creates a partition for each attribute that describes an instance. In each partition the instances are clustered according to their values for that specific attribute. Each partition is then evaluated according to its ability to discriminate between classes. C4.5 requires that any partition in the tree must have at least two outcomes with a minimum number of cases; in other words, the sum of the weights of the cases for at least two of the subsets must attain some minimum. The default minimum is two, but can be changed by the user via the weight parameter. The algorithm selects the best such partition and creates a child decision node for each cluster in the partition, associating the appropriate attribute test to the root node. When all the instances in the cluster belong to the same class the child decision node becomes a leaf, else the process iterates, passing on the instances in the cluster and the unused attributes, to build a sub-tree, with the current child decision node as the root. The process terminates once all the instances have been classified, or all the attributes have been used.

C4.5 learning algorithm's search is directed by two heuristic decisions. Firstly, the simplest, most compact decision tree must be generated and secondly, the decision tree must be consistent with the training environment, and have predictive powers. Quinlan (1994) uses the gain ratio criterion as an evaluation function to direct the search, in his quest to find the most compact, and consistent decision tree. The gain ratio criterion selects the attribute test that will maximise the ratio between the information gained, by partitioning a training set according to the attribute test, and the potential information generated by this partitioning. Therefore, the gain ratio expresses the portion of the information generated by the partitioning that is useful. The C4.5 learning algorithm simplifies its decision tree by means of an error-based pruning technique. This error-based pruning technique allows the algorithm to replace any one of its sub-trees by a leaf, or the most frequently used branch in the sub-tree, as long as it leads to a lower predicted error rate. A confidence factor is used to estimate this error rate. The default confidence factor is 25% but can be changed by means of the confidence parameter. Redundant complexes are finally pruned; a redundancy factor is used to identify these complexes (default 1.0). The two heuristic decisions based on gain ratio and predicted error rate, determine that the decision tree constructed during a search will be as simple and accurate as possible.

The C4.5 conversion function algorithm, C4.5rules uses the un-pruned decision tree generated by the C4.5 learning algorithm as a starting point. The algorithm rewrites the tree as a collection of rules. One rule is generated for every leaf, by tracing all the test outcomes along the path from the root to that leaf. A rule is induced as a conjunction of these tests. Rules are simplified by removing conditions that do not affect the accuracy of the rule, by means of a pessimistic accuracy estimate. The rules are then grouped according to their predicted classes and those that do not contribute to the accuracy of the rule set as a whole are removed. The rules for the classes are then ordered to minimise false positive errors and a default class is chosen.

### 2.5.6 BRAINNE– an artificial neural network learning agent with rule extraction algorithm

BRAINNE is an artificial neural network (ANN) generator that extracts rules from a trained neural network [Sestito 1994].

#### 2.5.6.1 Environment and performance task measures

The input to the system is both the training data set, as well as the structure of the defined inference network. The system generates an inference network, according to the specified structure. This trained inference network will be able to make the most accurate predictions about novel test instances.

The learning system, presented with one instance from the training set at a time, generates an inference network. Since the instances are presented to the learner one at a time it indicates that a form of online, supervised learning is occurring.

Once training is completed, the trained inference network will be able to classify an unseen instance according to the concept described by the network. In other words, the performance task of the BRAINNE environment is a form of classification. The performance of the network is measured by means of the mean squared error between all the original inputs and the generated outputs, over all the test instances presented.

#### 2.5.6.2 The knowledge base and performance element

As mentioned above, the input to the learning system is presented as a set of examples. Every instance in the set is represented by a set of numeric, continuous or discrete attributes describing the instance as well as the class it belongs too. Furthermore, the structure of the inference network must be specified in terms of the number of hidden layers and for each hidden, input and output layer the number of neurones must be specified. Also, the type of activation function, the learning rate and the momentum at which learning should occur must be defined for the learning system. BRAINNE uses the Sigmoid activation function.

The acquired knowledge of an ANN is represented in terms of a weighted value assigned to each connection strength between neurones. BRAINNE organises the output as a multi-layered neural network, also known as an inference network, consisting of neurones, input and output connections and weights. The generated inference network is the input to the rule extraction algorithm. The extraction algorithm then extracts concept descriptions describing the inference network. The concept description language describes the neural network in the format of Figure 6.

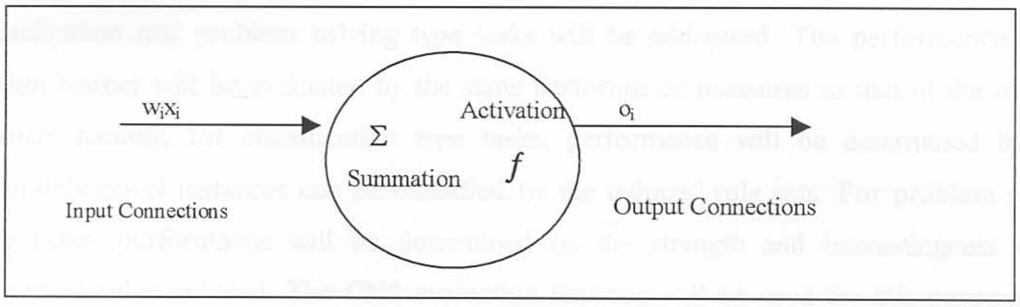

**Figure 6 BRAINNE concept description language**

### 2.5.6.3 The learning mechanism

The BRAINNE algorithm searches the complex space by means of an incremental gradient descent method. A linear threshold unit is induced for each class by means of the back propagation algorithm using the least mean square (LMS) as the evaluation function. The aim of the LMS is to minimise the mean squared error between the predicted and the actual classes by adjusting the weights assigned to each attribute. This is done in an iterative process, by specifying the number of epochs, passes through the training set, as well as the learning rate or size of steps through the weighted space. The process terminates once the number of epochs has been reached or the mean square error has decreased to a predefined level.

## 2.5.7 Human learner– learning agent

Viktor (1999) modelled a human learner as a learning agent using the inductive machine learner architecture. The sections that follow cast the human learning agent into Langley's machine learning framework.

### 2.5.7.1 Environment and performance task measures

The performance task of the environment that the human learner resides in is complex. The human learner must be able to acquire, generate, analyse, manipulate and structure information in order to construct knowledge [Alavi 1994]. For the purpose of this study only classification and problem solving type tasks will be addressed. The performance of the human learner will be evaluated by the same performance measures as that of the machine learners namely, for classification type tasks, performance will be determined by how accurately novel instances can be classified by the induced rule sets. For problem solving type tasks, performance will be determined on the strength and interestingness of the individual rules induced. The CN2 evaluation function will be used for this purpose. The human learner that participated in this study made use of both supervised and unsupervised learning, as defined in Section 2.5.1, by using the classified training set, but also ignoring the given classifications at times and generating new classes.

### 2.5.7.2 The knowledge base and performance element

The input to the human learner is presented as a set of training examples, identical to the set presented to the machine learners. The output of learning i.e. the knowledge acquired by the human learner is transferred into a knowledge base.

This process of capturing knowledge from human learners is referred to as knowledge acquisition. Turban (1995) defines knowledge acquisition as *"the process of extracting, structuring and organising knowledge from one or more sources"*. Knowledge acquisition does not concern only the extraction of acquired knowledge from the learner, but also concerns the transfer of that knowledge to a knowledge base. Many researchers have identified this process, as a bottleneck that hampers the development of artificial intelligence systems [Turban 1995]. The reason being, the nature of the knowledge possessed by humans is often unstructured and inexplicitly expressed which makes it nearly impossible to transfer into a machine-based knowledge base.

In every day life, transferring information from one person to another is difficult. Different media are often used, for example, spoken words, written words and pictures, yet none of them are perfect. The same problems that exist when transferring information from one person to another exist when transferring knowledge from a human to a machine. However, transferring knowledge from a human learner to a machine-based knowledge base is even more difficult. This task of transferring knowledge is the responsibility of the knowledge engineer. Turban (1995) defines knowledge engineering as *"a process involving the co-operation of human learners and a knowledge engineer in a problem domain, to codify and make explicit the rules that a human learner uses to solve real life problems"*. Sestito *et al* (1996) defines a knowledge engineer as *"the person that communicates with the expert to acquire the relevant knowledge"* and *"... looks at books, manuals, case studies and other material in order to better understand the problem domain"*. The knowledge engineer faces two major obstacles during the process of knowledge acquisition namely, knowledge expression and knowledge transfer. The knowledge that the human learner applies to solve a real life problem must be expressed as a set of rules. However, the process of solving a problem is an internal process. This requires the human learner to be introspective about his/her decision making process. The human learner is often unaware of this detailed process he/she uses to arrive at a conclusion and of the vast amount of previously acquired knowledge influencing his/her decisions. Hence, a different set of rules might be expressed to the knowledge engineer than what is actually used to solve the real life problem. The second obstacle, knowledge transfer, encountered by the knowledge engineer concerns transferring the acquired knowledge to a machine-based knowledge base. Machines express knowledge at a lower, more detailed, level than humans. Humans express knowledge in a compact form and are unaware of all the intermediate steps used by their brain in processing the knowledge. This creates a mismatch between human and machine representation.

Several methods for acquiring knowledge from human learners exist. For the purpose of this study the elicitation of knowledge from the human learner is done manually via structured interviews as described by Turban (1995). Structured interviews places great demands on the human learner who must be able to demonstrate his/her expertise in the problem domain, as well as be able to express it. However, structured interviews reduce interpretation problems and allow the knowledge engineer to prevent distortions caused by the subjectivity of the human learner [Turban 1995].

Once the knowledge engineer acquires knowledge from the human learner it is organised in a chosen configuration. A variety of knowledge representation schemes exist, for example, production rules, decision trees, semantic networks, decision lists and many more. All the schemes have two things in common, firstly, they can be stored in a machine-based knowledge base and secondly the knowledge base can be manipulated by a learning mechanism to perform an intelligent function. For the purpose of this study the knowledge acquired from the human learner will be converted into decision lists using a concept description language similar to that of CN2, as discussed in Section 2.5.4.2. Decision lists should be used as the knowledge representation scheme because the CN2 evaluation function is used to evaluate the performance of all the learners participating in the CILT-MAL system.

### 2.5.7.3 The learning mechanism

A mother of a one-year-old toddler observes that her child is fussy and irritable for several days. She decides that the child is teething. A one-year-old takes a bath every day, every time the child gets into the bath the mom says "Bath, bath" pointing to the bathtub filled with water. One day the child discovers a swimming pool in the garden. The child points and says "Bath, bath". These are examples of induction, where the learner learns from example [Johnson-Laird 1988] . Humans understand their environment by creating a simplification of the environment, called a model. A model consists of classes, representing similar objects in the environment, and class descriptions predicting the behaviour of the objects in the different classes [Holsheimer 1994]. The creation of such a model is called inductive learning, as discussed in Section 2.1.

Philosophers, psychologists and researchers of artificial intelligence have spent hundreds of person-years trying to characterise inductive learning, develop learning theories and attempt to program systems to improve themselves. Yet the current research has little in common with the way people actually think and learning theories often fail to represent human thinking [Holland *et al* 1986]. However, what we do know is that "*the human learner observes a series of objects and hence creates a hierarchy that summarises and organises experiences, building a prototypical example of each concept. As the human learner encounters new examples the prototype is refined*" [Gennari *et al* 1989]. Research conducted by Nisbett *et al* (1983) indicated that prior knowledge and experience, i.e. expertise in a problem domain, have a major impact on a human's inductive learning process. Those theorists that suggest experts typically do not reason using rules of formal logic but instead rely on memory specific experiences support this theory [Holland *et al* 1986]. The human learner that participated in this study used a combination of inductive learning and relying on his memory in areas of expertise. The human learner most often searched the problem space by means of a general-to-specific search, although this was not always the case. The human learner's searches were directed by heuristics based on the learner's previous experience within the problem domain.

## 2.6 Conclusion

Three different learner teams were used during this study, namely: A machine learner team that consisted of three machine learners, CN2, C4.5 and BRAINNE; A human learner team consisting of a human expert and a report, Technology base of large, medium and small organisations in the South African business sector [AMI 1998]; And a machine-human learner combination team that consisted of a human learner, CN2 and C4.5.

The CILT-MAL system consists of co-operative inductive learning agents that interact with the environment, by accepting training examples and feeding acquired knowledge back. The agents also interact with one another by sharing their individual knowledge bases. Using the inductive machine learner architecture, an agent can be divided into five conceptual components namely, the learning element, performance element, critic, data generator and knowledge base, as described in Section 2.3.

Langley's machine learning framework describes a learning system in terms of four components namely, the environment, performance task measures, learning mechanism and knowledge base. Hence, a learning system described by Langley's machine learning framework, relates to a learning agent, described by the inductive machine learner architecture in the following way: The environment of the one corresponds to the environment of the other. The performance task measures of the one correspond to the critic of the other. The learning mechanism of the one corresponds to the combined role of the learning and performance elements of the other and lastly the knowledge base of the one corresponds to the knowledge base of the other, as described in Section 2.5.

Within Langley's machine learning framework a learning system interacts with the environment but not with other learning systems. To relate Langley's framework to a multi-agent learning system, the individual learning systems must be able communicate with one another, as the learning agents do in a MAL system. A learning system has its own knowledge base in which it stores all the knowledge acquired during learning. Sharing their individual knowledge bases will enable the learning systems to communicate. However, this will only be possible under the following two conditions, namely; if the knowledge contained in the knowledge bases are represented in a uniform way across all the learning systems and if a uniform performance task measure is used across all the learning systems. Adhering to these conditions, a learning system will be able to interpret, as well as determine the quality of the knowledge acquired by a team member.

This chapter described co-operative inductive learning and then proceeded to define co-operative inductive learner teams. Co-operative inductive learner teams were then modelled as CILT-MAL systems. Within the CILT-MAL system learners were described using the inductive machine learner architecture. A suitable framework, namely, Langley's machine learning framework, was introduced. This framework was used to better define the learning mechanisms of the learning systems that participated in this study in terms of the key aspects of the framework's four components, which is summarised in Table 2, emphasising their main differences.

**Table 2 Key aspects of the learning agents modelled using Langley's machine learning framework**

| Learning agent | Performance task | Degree of supervision | Representation of output | Search technique | Search heuristics |
|---|---|---|---|---|---|
| CN2 | Classification | Offline supervised | Decision list | General to specific | Information-theoretic entropy measure<br>Significance measure |
| C4.5 | Classification | Offline supervised | Concept hierarchy | Greedy method | Gain ratio criteria<br>Error based pruning |
| BRAINNE | Classification | Online supervised / unsupervised | Inference network | Incremental gradient descent method | Least mean square error |
| Human | Classification<br>Problem solving | Online/Offline supervised/ unsupervised | Decision list | General to specific | Previous experience |

By focusing on these differences the performance of the different learning systems within the co-operative inductive learner teams, as presented in Chapters 4 and 5, will be better understood.

The next chapter, Chapter 3, describes the NRT Audit case study.