Learning Explainable Concepts in the Presence of a Qualitative Model

Thierry Rouget

Thesis submitted to
the School of Graduate Studies and Research
in partial fulfillment of the requirements
for the Master degree in Computer Science

The Ottawa-Carleton Institute for Computer Science

University of Ottawa

ISBN 0-612-11595-X

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

Acknowledgment

I would first like to thank my supervisor Stan Matwin for his help and availability, as well as for the interest he showed in my research. I also want to thank everybody who helped my stay at the Department of Computer Science of the University of Ottawa, and especially Jori Leonardon from l'Ecole des Mines de Saint-Etienne, who made the exchange program possible.

A lot of people helped me in the development of this thesis, and I would like to thank all of them, and particularly Peter Clark for his always enlightening advice and Antoine Bonavita for his availability and help with Prolog.

Finally, I want to thank Frederique for her constant support, especially during the last months.

Table of contents

Table of Figures

Chapter 1

Introduction

1.1 Overview

This thesis presents a method that uses possibly imperfect qualitative knowledge of a physical system to bias inductive learning. The expected benefits of this approach consist essentially in increasing the interpretability of the results. By trying to optimize the proportion of induced rules which are explainable with respect to a qualitative model of the physical system under study, we intend to ease considerably the interpretation of the results by a human expert. Moreover, explainability of induced rules helps with the possible integration of these results back into a knowledge base. We applied our method to three systems, a water tank network, an ore-grinding circuit and a small analog electric circuit, thus illustrating how our approach can be applied to practical process control problems. For each of these application domains, given the values of the observable parameters a time T, the learning task is to predict whether a parameter of interest, also called the target parameter, will have increased or decreased by time $T+\Delta T$.

Other work related to our thesis in various ways is presented in chapter 2. This includes systems mixing both explanation-based and inductive learning, a knowledge guided modification of C4.5 [Quinlan, 1993] and a medical expert system which was developed on the basis of a qualitative model of the heart.

In chapter 3, we formally define our knowledge representation, i.e. we describe our adaptation of the qualitative models of the Qualitative Process Theory [Forbus, 1984]. A major difference of our models, as opposed to models usually developed in Qualitative Physics, lies in the fact that they are not intended to be able to actually simulate the behavior of the physical system under study. Rather, we use qualitative models to capture our qualitative knowledge of the system, and to determine whether a given prediction rule is consistent with this knowledge or not. We thus define what it means for a rule to be explainable by the qualitative knowledge captured in the model, and describe how we extract all the rules which are explainable with respect to the qualitative model and store them so that they can be efficiently used during the inductive process.

The learning algorithms we propose are modifications of CN2 [Clark and Niblett, 1989; Clark and Boswell, 1991] which take into account whether rules considered during the inductive search are explainable or not. In chapter 4, we present a quick overview of CN2 and of our first method, the totally constrained search, which restricts the space heuristically searched during learning to the space of explainable rules.

Chapter 5 contains the presentations of two other methods, whose biases for explainability are weaker than the bias of the totally constrained search. These methods are intended to be able to cope with imperfections in the qualitative model. This goal is achieved by allowing non-explainable rules to be induced when the restriction of the search space to explainable rules has strong negative effect on the classificational accuracy of learned rules. For example, if a model is severely incomplete, the resulting space of explainable rules will be greatly reduced, resulting in a possible loss of predictive accuracy. We thus expect these methods to allow a practical trade off between the high accuracy of the results of a pure inductive learner like CN2 and the explainability of the rules learned by the totally constrained search. In addition, we show that the first of these two methods can be used to suggest completion of an incomplete qualitative model based on statistical evidence found in the training data. As for the second method, it allows a representation of

uncertainty in the qualitative model to be taken into account during learning. It could also be used to enhance the quality of an initial model , obtained using the first method.

Chapter 6 is devoted to an experimental evaluation of the three methods proposed in chapters 4 and 5. We first compare the totally constrained search to the standard CN2 algorithm, when a reasonably good qualitative model of the physical system is available. We also evaluate our two other methods by providing them with incomplete models.

Chapter 7 concludes this thesis and summarizes what, we believe, are the main achievements of our research. In this chapter, possible extensions and future work are also discussed.

1.2 Motivation

It has been often reported that applications of purely inductive tools like ID3, C4.5 or CN2 to real-world tasks involves some background knowledge. In most practical applications, results of the inductive learning are shown to domain experts to see if they 'make sense'. This revision of the empirically acquired knowledge can lead to modifications of the induction procedure like the removal or adding of sensible training examples, modifications of the example description language, or direct modifications of the learned decision tree or list of rules by the knowledge engineer. A new version of the results is then presented again to experts, and the process iterates until the resulting set of rules (or decision tree) is both accurate and acceptable by experts.

This process thus illustrates an interaction between results that were purely empirically obtained and the knowledge of domain experts. This has the main disadvantage of being time-consuming. A preferable solution would be to integrate domain knowledge into the inductive tool itself to generate results with are both accurate and consistent with the domain knowledge. Also, as often in machine learning applications ([Ortega and Fisher, 1995]), the introduction of domain knowledge can possibly enhance the classificational accuracy of the learned rules, by pruning out rules which coincidentally perform well on

training data but which are inconsistent with domain knowledge. If the domain knowledge is perfect, these rules will probably have a poor accuracy on unseen data and will degrade the overall accuracy of the system.

In this thesis, we present a method for applying qualitative knowledge of a physical system to guide inductive learning. The first approach restricts the choices available to the inductive engine to the subspace of rules which are explainable with respect to a qualitative model of the physical system under study. Expected benefits are a higher interpretability of the learned prediction rules, since they can be explained by the qualitative knowledge provided to the system, as well as a possible gain in accuracy. In our implementation, we indeed generate, for each rule of the learned ruleset, a qualitative explanation of the causal reasoning in the qualitative model which explains this particular rule.

However, when the qualitative model is imperfect, it might not be possible to obtain accurate results by restricting the search space to explainable rules only. In this case, the best choice seems to try to maintain high accuracy by allowing non-explainable rules to be learned if they have a higher predictive power. The learning task thus becomes to learn rules with high accuracy while also trying to optimize the proportion of explainable rules in the induced ruleset. As an alternative to the first method for the case when the only available qualitative model is severely imperfect, we propose another method which trades off between accuracy and explainability of the results.

Furthermore, as also commonly reported when developing an application of the inductive technology to real-world problems, strong statistical evidence found in the data may cause experts to revise their domain knowledge. We exploit this aspect of knowledge acquisition by suggesting revisions of imperfect qualitative models based on strong correlation in the training data.

Finally, we propose a representation of uncertainty which is intended to help experts to build qualitative models of physical systems for which the domain knowledge might be

poor or not totally reliable. This representation is based on the addition of belief weights in the qualitative model. A belief weight denotes the level of confidence the model builder has in a particular link. We also define, by convention, that a link not in the qualitative model has an associated belief weight equal to zero. This allows our last method to overcome imperfections in the qualitative model by learning non explainable rules if the accuracy improvement is high enough to override the default bias favoring rules with high belief values. In this case too, when the model is severely incomplete or incorrect, we expect to allow a practical trade-off between the accuracy and the explainability of the results.

1.3 Contribution

The research presented in this thesis is an extension of a previous work by Peter Clark and Stan Matwin. Principles of the knowledge representation and a first approach assuming a perfect qualitative model (corresponding to chapter 3 and to the totally constrained search of chapter 4 in this thesis) have been already introduced in [Clark and Matwin, 1993]. However, the notion of explainability of a rule, with respect to a qualitative model, was made more precise in our research and is formally defined for the first time in chapter 3. We also present a new application domain, which is related to process control of an analog electric circuit. These two factors motivated a new experimental evaluation of the totally constrained search, comparatively to CN2, which is described in chapter 6.

In addition to these improvements, the contribution of this thesis consists mainly in the two new approaches presented in chapter 5, which are intended to deal with possible imperfections in the qualitative model. Means of feedback of the learning results into the qualitative model, allowing revisions of imperfect models, are also presented as a feature of the first of these two methods. Furthermore, we show in the second part of chapter 5, how uncertainty can be represented at the qualitative model level, and advantageously exploited during search. Finally, in chapter 6, we present an experimental evaluation of

these two new methods and show how they are an improvement over the totally constrained search.

Chapter 2

Context and Related works

2.1 Context

In knowledge acquisition techniques, two main general approaches can be distinguished. The first one is to directly encode the expert's knowledge into an operational form for prediction or diagnosis. Since these methods were developed to build the first expert systems, they are often called traditional knowledge acquisition strategies. An alternative is to use machine induction from data. In general, the choice between these two methods depends on the availability, size, representativeness of data as well as the quality of human expertise in the domain. If the data is plentiful and the expertise is not strong, then machine induction seems to be a natural choice.

While earlier inductive systems generally relied exclusively on data to generate the needed concept description, there is now a growing interest for approaches that combines model-based or domain-theory knowledge with empirical learning. There are two principal ways in which background knowledge can be used:

- to expand the hypothesis language. In this case, the knowledge provided is used to introduce new terms (e.g. in FOCL [Pazzani and Kibler, 1992]).

- to guide and constrain search (e.g. [Evans and Fisher, 1993]).

Generally, the introduction of background knowledge in an empirical learner is motivated either by an increase of predictive accuracy or by the interpretability of the results. In the

first case, domain knowledge is intended to help the learning and increase its efficiency. FOCL [Pazzani and Kibler, 1992] is an example of a case where even partially correct domain-theories improve the predictive accuracy. In our research, the motivation for background knowledge is mainly interpretability of the induced rules. We use a qualitative model to represent the behaviour of the physical system under study and to both guide the search and generate qualitative explanations of the induced rules.

Moreover, background knowledge may not be perfect and data may be noisy. As a result, the data may conflict with the domain-theory. Some methods assume in this case, that the data is noisy or drawn form a very narrow subspace of the data and is thus not representative. Other methods rely more strongly on data and even use information found in the data to correct or complete the original domain-theory. These approaches are known as the theory revision methods. EITHER [Ourston and Mooney, 1994], for example, is able to learn from overly specific or overly general (and possibly incorrect) theories and also to revise them. While it is possible to use our approach to revise background knowledge in a limited manner, theory revision is not the main goal of our work. Our principal objective is to try to maintain a high accuracy and explainability of the induced ruleset, even when the domain theory is incorrect or incomplete. The notion of explainability of a rule with respect to background knowledge is introduced in chapter 3.

An alternative to these two strategies is to couple more flexibly empirical learning and model-based reasoning. In the next sections, we will present three methods that integrate both explanation-based and inductive features. We will also present KARDIO, which is an expert system built from a qualitative model with the use of machine learning techniques.

2.2 ML-SMART

In the field of automated acquisition of concept description, there are two main approaches that can be distinguished: empirical learning and explanation-based learning (EBL). The first approach uses mostly induction and consists in generalising examples to produce a concept description. The second approach uses deduction from a domain

theory, a non operational concept description and one training example, to produce an operational concept description.

However, one strong limitation of the second approach is that it often requires that the domain theory is perfect. An alternative to these two approaches is to combine both and use deduction and induction in an integrated way. ML-SMART [Bergadano and Giordana, 1988] is an example of such a combination.

ML-SMART uses data, domain knowledge and tentative concept descriptions to produce a discriminant operational concept description. Deduction from a non-operational domain theory is interleaved with data-driven inductive steps. The inductive search is intended to overcome some limitations (inconsistency or incorrectness) of the domain theory.

The general strategy of ML-SMART is a best first search for specialisations of a non-operational concept description. The search begins with the tentative concept description (which can be either non-operational, partly operational or totally operational) given by an expert. First, ML-SMART adopts a pure EBL behaviour and tries to deduce from the domain theory the needed expansions. Note, however, that the whole learning set is used rather as opposed to a single example in the classical EBL approach.

If the theory is imperfect, then the explanation-based operationalization of the goals can fail. In this case, ML-SMART tries to recover by using induction of some incomplete or inconsistent proofs. After a few inductive steps have been performed, the possibility to continue with deduction is considered. If deduction fails, then only induction is used to continue the search.

This work presents some similarities with our approach. The qualitative model we use can be seen as ML-SMART's domain theory. However, there is a fundamental difference in the way each system uses its background knowledge. We use the qualitative model to guide an inductive search, not for deduction. There is no explanation-based learning in our approach. Nonetheless, we will see in section 5.1, that the method we developed to deal

with imperfect theories resembles to the one found in ML-SMART. When theory-guided search seems to perform badly, then we rely on pure induction (without background knowledge) to continue.

2.3 FOCL

Like ML-SMART, FOCL [Pazzani and Kibler, 1992] is a hybrid explanation-based and inductive learner. It also learns relational concept definition based on constant-free Horn clauses, but without the strong restrictions ML-SMART places on the form of induction and the initial knowledge that is provided to the system.

FOCL (First Order Combined Learner) is an extension of FOIL ([Quinlan, 1990]), in which various types of background knowledge are added. In FOCL, background knowledge can be typing information, multiple argument constraints, both extensionally and intensionally defined predicates (the latter are the domain theory), as well as an initial, approximate definition of the concept to be learned.

As special cases, FOCL can behave like a pure explanation-based learner or like a pure inductive learner. With a correct and complete domain theory, the output of FOCL is similar to that produced by m-EBG ([Flann and Dietterich, 1989]). With an empty domain theory (no intensionally defined predicates), FOCL operates like FOIL. When given an imperfect background knowledge (an incomplete or incorrect domain theory), clauses of a rule may be learned purely analytically, purely empirically, or by a combination of both methods.

FOCL uses FOIL's information-based metric (see [Quinlan, 1990] for a complete definition of the information-based metric) to evaluate extensions to a (possibly null) hypothesis of a concept definition. The same metric is thus used to evaluate the pertinence of extensions provided by pure induction, suggested by intensionally defined clauses of the domain theory, as well as the pertinence of using a complete (or a subset of a) tentative concept description given by an expert. This is a major difference with ML-SMART,

which uses a number of statistical, domain independent, and domain dependent heuristics for selecting whether a rule is to be extended by inductive or deductive means.

FOCL performs an hill-climbing search guided by its information-based heuristic. In contrast, ML-SMART with its best-first search can solve problems which cannot be solved with hill-climbing alone, but at a higher computational cost. ML-SMART is thus limited to smaller problems. Furthermore, intensionally defined clauses (i.e. clauses defined in terms of extensionally defined predicates) allow FOCL to take wider steps during its hill-climbing search, and thus improve the hill-climbing landscape. These clauses allow FOCL to solve problems which cannot be solved with smaller steps, due to local maximums. On the other hand, intensionally defined clauses increase significantly the search space.

Also, background knowledge is not only used for explanation-based reasoning like it is in ML-SMART. In FOCL, inductive learning is guided by some forms of background knowledge. For example, predicates have typing information added for the arguments. The predicate *greater_than*(X,Y) specifies clearly that X and Y are variables of the type numbers. Thus, it prevents extensions with variables of other types to be considered during inductive search. Of course, typing information is not considered during explanation-based learning, since it is implicitly assumed that the domain theory itself will not violate typing constraints. Multiple argument constraints (like "the variables in this predicate have to be different") are also provided to prevent trivially true extensions like *equal*(X,X) to be considered during the search. Both typing and multiple argument constraints reduces dramatically the search space and are thus very useful. Typing can also improve the accuracy of the hypothesis by eliminating inconsistent extensions from the search space.

Similar to FOCL's inductive approach, this thesis presents a way to use background knowledge to guide and constrain a hill-climbing inductive search. Our domain theory is a qualitative model of the physical system under study. During search, candidate rules (i.e.

rules considered during the inductive process) might be explainable or not with respect to the knowledge captured by the qualitative model. We then define high level strategies that decide how the explainability biases the search. One of these strategies discards rules which are not explainable with respect to the qualitative model and is comparable with FOCL's typing constraint. We also expect the same benefits, i.e. to reduce the search space and thus to speed up the learning and also to improve accuracy by pruning out non explainable rules which coincidentally perform well on training data.

Another interesting feature of FOCL is that it learns more quickly with a domain theory than without any background knowledge, even if the domain theory is only partially correct. Here, more quickly means that less training example are required to reach a given accuracy for the learned concept description.

2.4 Flexibly exploiting prior knowledge in empirical learning

[Ortega and Fisher, 1995] presents a modified C4.5 ([Quinlan, 1993]), which takes into account background knowledge. In this approach, data is augmented by features that can be seen as terms of a domain theory. For example, in the example of the classifier of faults of the Reaction Control System of the Space Shuttle, a previous mixed qualitative and quantitative model was available. The terms of the domain theory represented by this classifier are the model's prediction, as well as intermediate computations. Thus examples are added attributes, whose values are the corresponding values of the class, as predicted by the previous classifier, and of some of the intermediate computations used by the latter. Induction is then performed over this augmented data set.

The originality of Ortega and Fisher's work lies in the fact that, during induction, C4.5 is biased to select domain theory features, even when this conflicts with C4.5's original bias. Without this additional bias, C4.5 would simply select the most informative feature as computed over the data. The intent of the bias in favour of the domain theory features is to guard against noisy or unrepresentative data. However, this bias is overridden when the

information-based bias is significantly opposed to the domain theory bias. Here, the intent is to acknowledge that the domain theory might not be perfect.

Model features are classified according to their abstraction level in the domain theory. Level 0 (the highest abstraction) is the model prediction itself, while level 3 is represented by the raw data features. Level 1 and 2 represents intermediate concepts (like for example a conjunction of raw features) which are used in the domain theory to generate the prediction.

During induction, the domain theory bias makes C4.5 select the features of highest level unless a feature of lower level is significantly better in term of the information gain ratio. To determine whether a feature is significantly better than another one, the hypothesis that the information values of the two features are the same is tested statistically. If this hypothesis can be rejected with a high enough confidence level, then the best hypothesis is said to be significantly better.

Experimental results show that the accuracy of the learned classifier increases with the quality of the domain theory. Also, the domain theory bias with the significance testing makes the system more robust when confronted with skewed data. Finally, even imperfect (but still reasonably good) models result in an increase of accuracy.

This approach presents many similarities with our research:

- background knowledge is used to guide and to bias an inductive search.

- the domain theory bias can be overridden by a sufficiently high statistical evidence found in the training data.

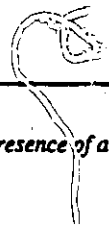
The threshold algorithm we present in chapter 5 can be seen as a special case of this learning paradigm, where the features of the domain theory are the rules which are explainable with respect to the qualitative model. The default bias selects these rules unless some statistical condition derived from the data shows that they perform poorly.

However, there are two major differences between the modified C4.5 presented in this paper and our research. The first one is that we use the background knowledge to restrict the search to a subspace. In contrast, the presented work introduces new terms derived from the domain theory and thus aggravates the search by expanding the space of possible concept descriptions. The second one lies in the way the modified C4.5 determines a significant difference. Since the form of the distribution for the information gain ratio is not known, *bootstrap methods* [Efron and Gong, 1983] are used to try to reject the null hypothesis that the two features have the same information values. In our research, we only compare the values given to the candidate rules by our heuristic to a given threshold. If all the candidate rules have a value below the threshold, then the domain theory bias is dropped. While less robust, this simple method saves a lot of computations, since bootstrap methods are particularly computationally expensive. Since no measures about learning times (in terms of CPU utilisation) were given in the paper, it is unclear in what proportions the search is slowed down by the use of the significance testing.

2.5 KARDIO

KARDIO [Bratko et al., 1989] was a project for building an expert system for the ECG diagnosis of disorders of the heart, known as cardiac arrhythmia. It involves a qualitative model of the heart, as well as machine learning techniques like induction.

An interesting aspect of KARDIO is that it distinguishes two levels of background (or expert's) knowledge, the *deep* and *surface* knowledge. The deep knowledge would be what is usually called the domain theory and contains laws that allow reasoning. In contrast, surface knowledge directly "states the relation between problem specification and problem solution without referring to the underlying principles". An example of surface knowledge can be prediction rules, where only some attribute tests are used to predict the class to which the example belongs. In surface (or shallow) knowledge, the reasoning involved in the prediction is hidden. Surface knowledge, on the other hand, is more



operational in the sense that it can be used to solve problems efficiently, since it is usually derived from deeper knowledge for a specific task.

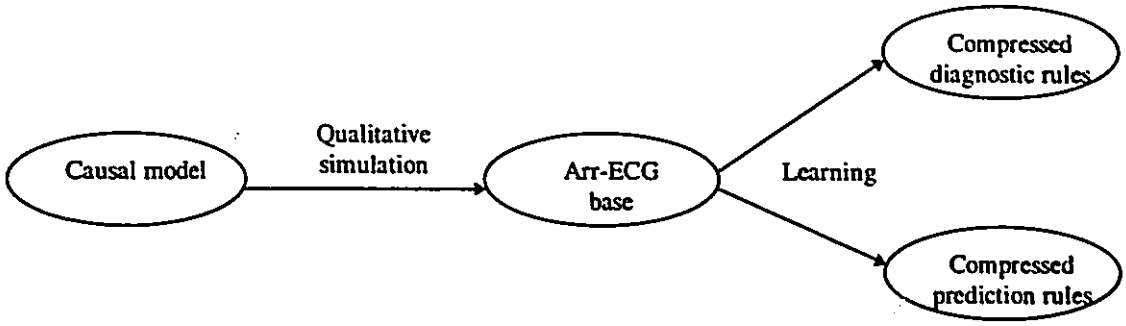


Figure 2.1 Knowledge representation levels in KARDIO

Figure 2.1 ([Bratko et al., 1989]) shows the different levels of knowledge in KARDIO and transformations between them. The deep level of knowledge is the qualitative model of the heart. This qualitative model is then used to produce through simulation all the possible combinations of cardiac arrhythmia along with their ECG descriptions. These ECG descriptions cover entirely the space of ECG descriptions. Since they are not very interpretable, mainly due to their size, a more compact description of the relation between ECG descriptions and the cardiac arrhythmia is extracted by inductive means by using the output of the simulations as the training examples. The final (surface) knowledge representation are compressed diagnostic and prediction rules. The compressed representations thus obtained are compact and largely comprehensible. These representations are thus more suitable to be compared by human experts to other knowledge codification of the same domain. In some cases, the compressed rules will disagree with common experts' knowledge of the domain and might allow corrections of this knowledge. Knowledge acquisition is thus a cycle as shown in Figure 2.2.

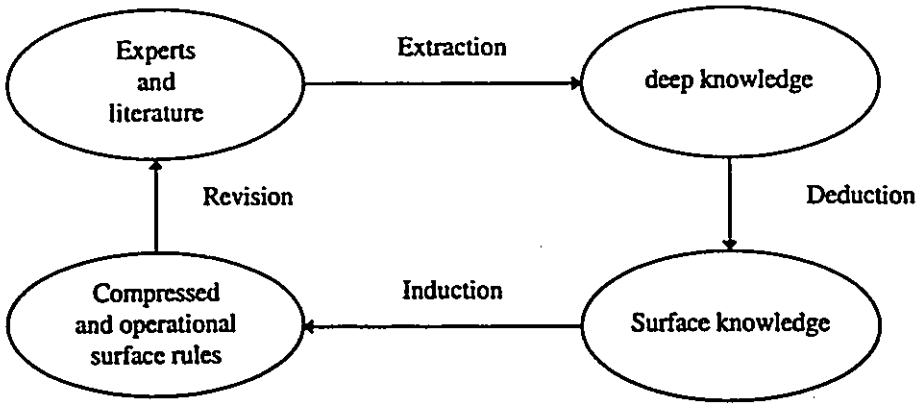


Figure 2.2 Knowledge acquisition cycle

KARDIO illustrates well the standard knowledge acquisition techniques and includes interesting features. First, instead of extracting only surface knowledge from experts and literature, deep knowledge was captured and represented as a qualitative model. This representation is very comprehensible and thus easily rectifiable. Also, this deep knowledge was used to generate examples for inductive learning. Since the examples cover the whole space of cause-symptom description, the induced rules will contain all the knowledge captured in the domain.

Our approach present similarities with the one presented in KARDIO. We also use a qualitative model to capture deep knowledge and we deduce from the model surface knowledge in the form of explainable rule schemas (see chapter 3). However there are two fundamental differences between KARDIO and our system. The first one is that our qualitative models are incompletely specified and cannot be used for simulations. Rather, the knowledge captured in the model is extracted in an automated way and used to guide the inductive process on independent data. In KARDIO, the model is not used to constrain induction. The second difference lies in the training data itself. Our system uses independent data and background knowledge to produce prediction rules. In contrast, KARDIO generates the training data from the background knowledge. In other words, in KARDIO, induction is just a way to create a more compact and more operational representation of the domain knowledge. All the knowledge necessary to the problem

solving is already present in the model. Our approach consists in extracting knowledge from data by knowledge-guided induction, and is thus fundamentally different.

However, KARDIO emphasizes the importance of comprehensibility of the surface knowledge in the knowledge acquisition cycle. Without any deep knowledge behind, surface knowledge cannot be easily interpreted. In our system, we also use background knowledge to generate qualitative explanations of induced prediction rules.

Chapter 3

Knowledge Representation

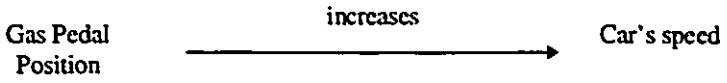
This whole work can be viewed as developing and evaluating a special case of the theory-guided learning paradigm we described earlier [Bergadano and Giordana, 1989, Pazzani and Kibler, 1992, Ortega and Fisher, 1995], in which the domain theory is a qualitative model of the studied physical system and the learning is rule induction from data. In this chapter we will describe how the background (or domain) knowledge, i.e. the qualitative model, will be integrated into the inductive process.

3.1 The qualitative model

3.1.1 Description

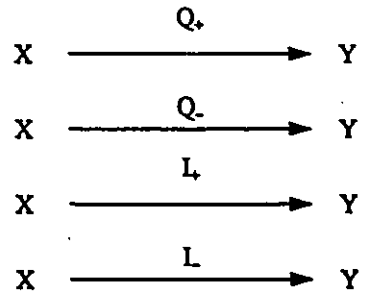
We use here the same terminology than Forbus in his Qualitative Process Theory [Forbus, 1984]. The physical system under study is composed of **objects**, which interact through processes. Each object has a set of continuous attributes or properties, also called **parameters**. An example of parameter is the mass of a car, the volume of a water tank, etc.

The qualitative model is a qualitative description of the behavior of the physical system. This description is a graph whose nodes are parameters of the application domain and whose arcs represent relationships between parameters (arrows indicating temporal precedence). For example, if we study the general behavior of a car, then we will probably introduce a description for the interdependence between the position of the gas pedal and the speed of the car, which looks like:



The diagram indicates that depressing the gas pedal will cause the speed to increase (if the engine is started, of course!).

The arcs we use are the same as those used in the Qualitative Process Theory [Forbus, 1984], i.e. either I+, I-, Q+ or Q-.



The Q+ (qualitatively proportional) relationship stands for: "there exists a function f that determines Y, and is monotically increasing in its dependence on X". In algebraic notation, we would write

$$Y = f(..., X, ...)$$

In others words, it means that if X increases, then Y increases and that if X decreases, then Y decreases too.

The Q- relationship also means that the two parameters are qualitatively proportional but with f monotically decreasing in its dependence on X. Thus, it means that if X increases, then Y decreases, etc...

The I+ relationship means that "X is a cause for Y to rise" or that the rate of change of Y (dY/dt) varies monotonically with X. In algebraic notation, we would write

$$dY/dt = f(..., X, ...)$$

with f monotically increasing in its dependence on X. Thus, it means that if X exceeds a given value then Y increases (or will increase). In the following, "X is high" actually refers to "X is greater than a given value". Thus, "X is high" should not be interpreted as "X's value is high in the range of X's possible values". This simplification is introduced to make qualitative explanations of the system's behaviour shorter.

The I- relationship also means that X is a cause for Y to rise but with f decreasing monotonic in its dependence on X. X being high causes Y to decrease.

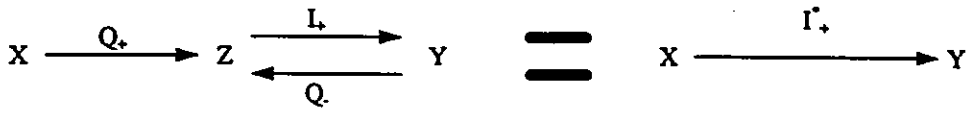
I and Q links are essentially different. If two parameters X and Y are connected by a Q+ link, knowing the value of X gives us information on Y's value. Furthermore, if we know that X is increasing, then we also know that Y is increasing. However, knowing X's value does not give us any information on Y's future change. For an I+ link, knowing the value of X tells us whether Y is increasing or decreasing, but it does not give us any information on Y's current value.

In our qualitative models, we will also use a third label I* in addition to I and Q to denote a self-stabilizing feedback loop. This link is not included in Forbus' Qualitative Process Theory. An I*+ link means that if X is high, then Y will rise until it reaches a new constant value. Returning to our example of the gas pedal of a car, it would be more judicious if an I*+ relationship was used to connect the gas pedal position and the car's speed since, in reality, what happens is that a depression of the gas pedal causes the speed to increase until it eventually reaches a new, higher, constant value.

In the same manner, an I-* relationship denotes an inverse monotonic relationship.

For short time scales, an I*+ link behaves like an I+ relationship, while for long time-scales, it behaves like a Q+ link (in the example of the car, every gas pedal position

eventually produces a corresponding speed). Also the figure below shows it is possible to decompose an I* link into I and Q links.



This can be intuitively understood by considering again the example of the car: X is the gas pedal position, Y the speed of the car and Z is the acceleration. The acceleration varies proportionally with the gas pedal position (hence the Q+ link). The change of the speed of the car is also determined by the acceleration: high acceleration causes the speed to increase quickly (the I+ link from Z to Y). But, we understand that the car's speed does not increase infinitely. Rather, what happens is that the air resistance increases with the speed and makes the acceleration decrease (the Q- link from Y to Z), because the acceleration is actually the engine force minus the air resistance. As a result, the speed increases slower until it reaches an equilibrium value when the acceleration is equal to zero. The I*+ link summarizes these three relations into one and hides the intermediary variable Z. By re-introducing a hidden parameter, it is always possible to decompose an I* link into Q and I links. Thus, rather than an addition to the language used to write qualitative models, the I* relation is a syntactic shorthand.

While very similar to the Qualitative Process Theory models, one should note that our qualitative models present a slight difference: they are "incompletely specified". We do not define any "quantity space" for each parameter, so we are unable to solve conflicting influences during simulation. Thus, the main difference is that our models on their own cannot be used for simulation or prediction. Instead, their role is to describe which processes we expect to take place in the physical system and to constrain induction to rules we can explain with respect to these processes.

In fact, the qualitative model represents the space of relationships which are considered credible in the domain by the model constructor. Note also that some of the parameters

included in a qualitative model may not be observable, i.e. their value was not measured or could not be measured during simulation. In the following, we will indicate by '**observable parameter**' a parameter whose value can be measured and was indeed measured to be included in the data sets.

3.1.2 Examples of qualitative models

Our method has been applied to three different domains related to process control problems: a water tank system containing feedback, a real-world process of ore-grinding, in which rock is crushed into smaller particles for mineral extraction, and a small analog electrical circuit containing operational amplifiers. For each of these systems, there is a parameter of interest (also called target parameter) whose movement we wish to predict. To generate data, numerical simulators (not the qualitative models) of the real physical processes were used or constructed.

3.1.2.1 The water tank system

The water tank system is shown in Figure 3.1. Water enters the system through the upper pipe and into the first water tank T1. From tank T1, water may flow into T2, T3 or T4 and so on. In addition, there are control valves which may prevent water from flowing through some pipes (for example, there is a control valve on the upper pipe). These valves are controlled by the level of the tanks. For instance, the control valve on the pipe between T1 and T3 is closed whenever the level in tank T5 becomes greater than a given value, but remains open when T5's level is below this value. These control valves are means of feedback from a tank to the water flowing into it (more or less directly).

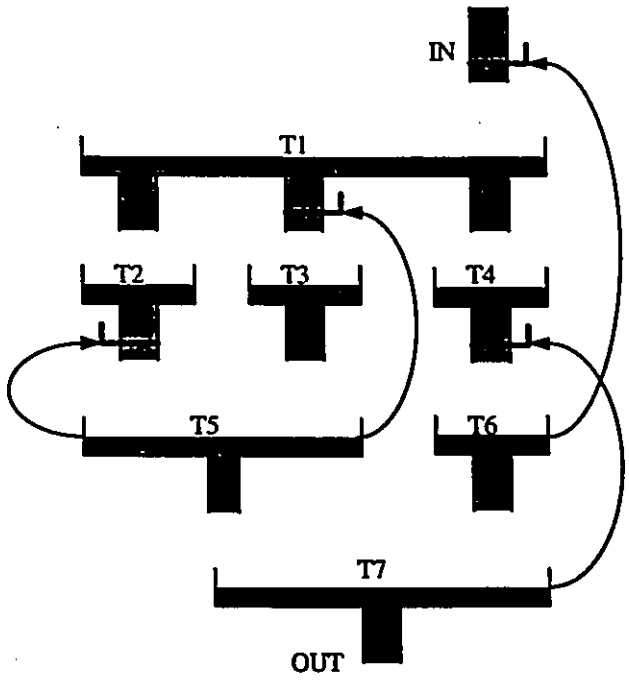


Figure 3.1 The water tank system

The only user-controllable parameter is the water flow entering the system through the upper pipe. The parameter of interest is the level of the water tank T7 (the tank at the bottom of the circuit). The level of each water tank is observable so the learning task is, given the levels L1, L2, ..., L7 of the tanks at time T, to predict whether the level L7 will have increased or decreased by time T+ΔT.

The qualitative model was based on the fluid dynamics law, which states that the flow of water out of a tank is increasing monotonically with the tank's water level. This is a direct qualitative translation of Bernoulli's equation:

$$V = \sqrt{2gh}$$

where g = 9.81 $m.s^{-2}$, h is the height of liquid in the tank and V is the speed of the flow. The higher the level, the greater the pressure at the bottom and the faster the out-flow. Hence, since the diameter of the pipe through which water flows out is constant, the faster

the out-flow, the bigger the volume of water that flows out of the tank. As a consequence, we expect the level in a water tank to increase with the level of water of a tank just above, as the out-flow from the latter also increases with its level.

Thus, if the level in tank 3 is high, then the flow of water from tank 3 into tank 5 will be high and will cause the level in tank 5 to rise, and we can deduce:

$$L3 \xrightarrow{\;I_+\;} L5$$

This cannot be modeled by a Q+ link. If we used a Q+ relation to connect L3 and L5, then it would have meant that "if L3 is increasing then L5 is increasing too". Here, what the physical law states is that knowing the current value of L3 tells us about L5's future change, which is different. Depending on the time-scale in use, this effect could also have been modeled by an I* link. Since when the level in tank 5 increases, the flow of water out of tank 5 increases too, so we expect the level in tank 5 to eventually reach a new equilibrium value. But, as we have already noticed earlier, if the time-step is relatively small, it does not make a significant difference.

Figure 3.2 and Figure 3.3 show two qualitative models of the water pipes system. The first model comprises only I links (no I* links) and is similar to the one presented in [Clark and Matwin, 1993]. The second model uses I* to summarize the loops and is thus a little

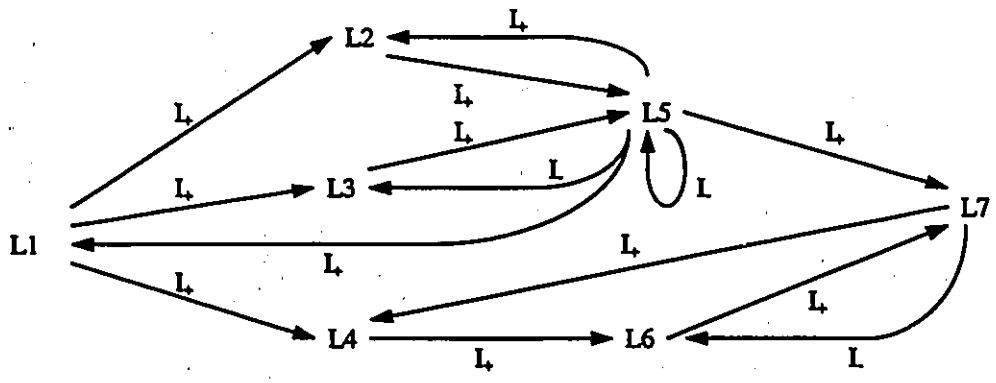


Figure 3.2 Detailed qualitative model of the water tank system

smaller.

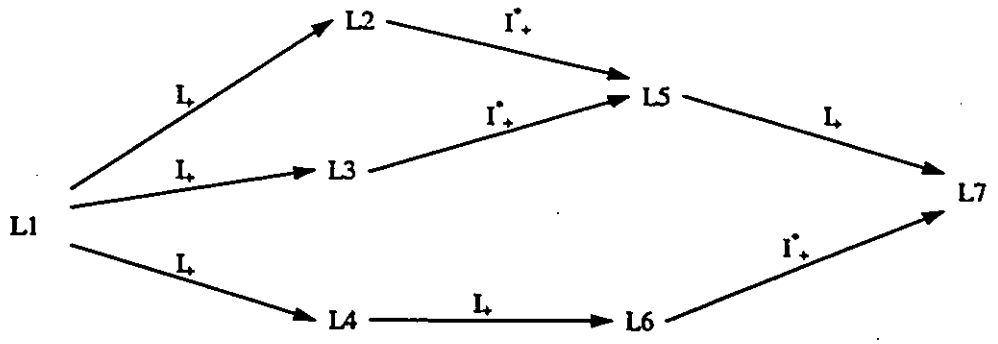


Figure 3.3 Qualitative model with feedback loops summarized

A simple numeric system written in Prolog was used to mimic the system's behavior with time and to generate data.

3.1.2.2 The ore-grinding circuit

The ore-grinding circuit shown in Figure 3.4 is a simplified version of a similar one used in the mining industry [JKTech Ltd., 1991]. Ore first enters ball mill 1, where it is crushed into smaller components (a ball mill is a large rotating drum whose function is to crush rocks into smaller pieces). A fraction of the ore then leaves the first ball mill and is conveyed to a metal screen. The larger rocks, those which cannot pass through the screen, are removed and fed back into ball mill 1. Ore that goes through the screen enters and accumulates in a large centrifuge, also called a cyclone. The smaller contents of the centrifuge are filtered out and leave the system (this is the output). The larger contents of the centrifuge are eventually removed and conveyed to a second ball mill, where they are crushed further. Ore leaving ball mill 2 returns to the cyclone. Water can be added to the ball mills to increase the out-flow, but it also decreases efficiency since a fraction of the energy is then used to 'grind' water.

There are four operator-controllable parameters, namely: the feed rate, the size distribution of the ore in the system and the rate of water addition into the two ball mills. The target parameter is the overall efficiency, which is the produced volume divided by the power used to produce it.

The qualitative model we constructed of this process is shown in Figure 3.5. It is meant to be a reasonable qualitative description of the physical system behavior but is, to a large extent, Clark and Matwin's guess.

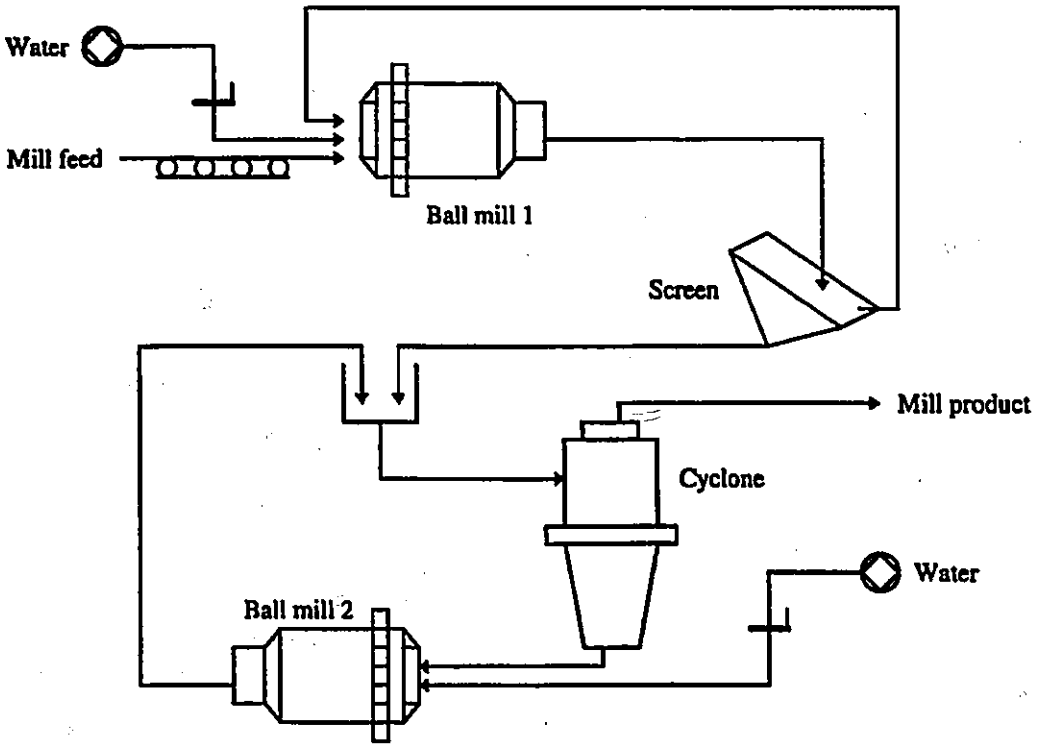
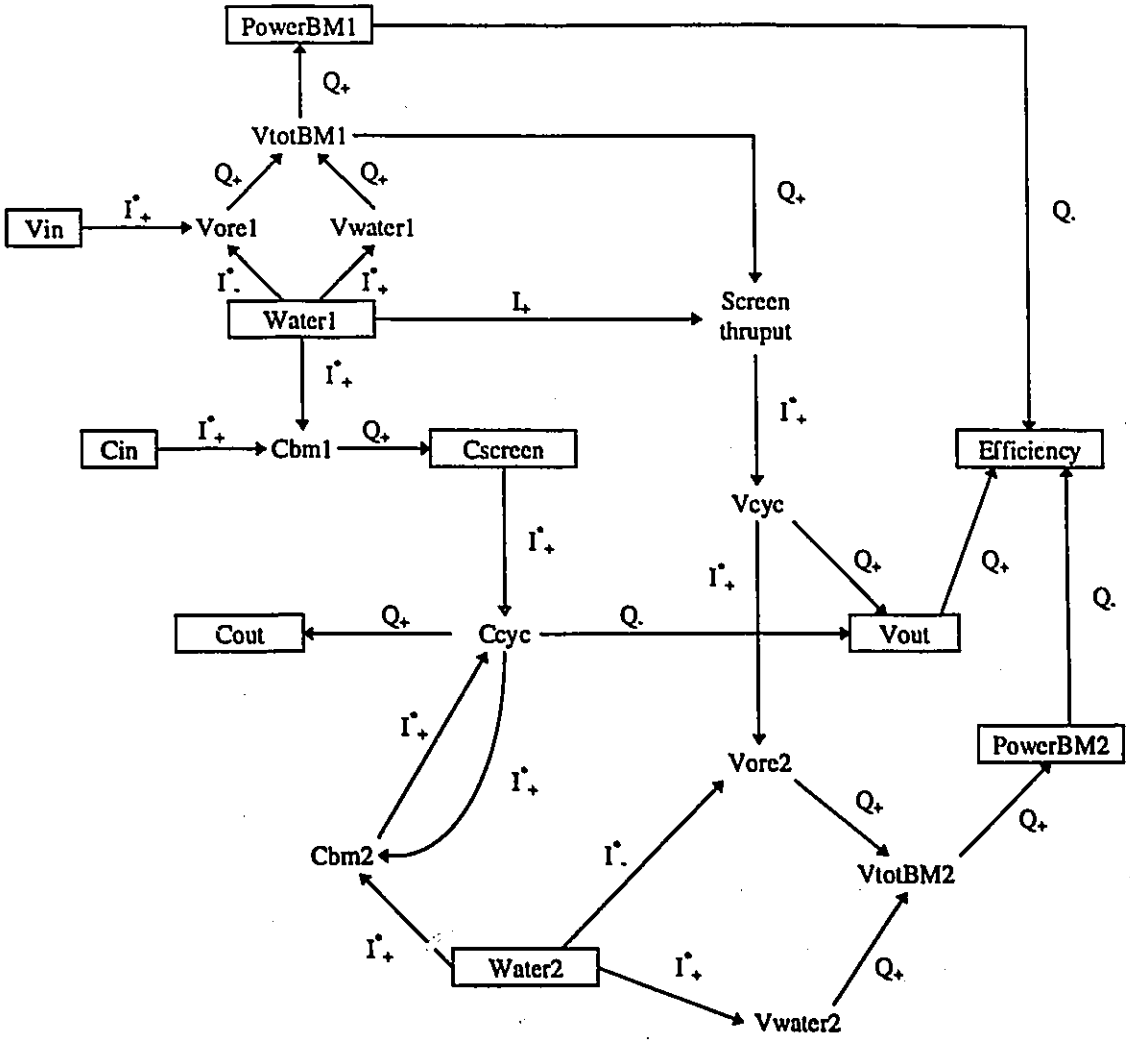


Figure 3.4 The ore-grinding circuit



Qualitative model of the ore grinding circuit

Figure 3.5

The 10 observable parameters, shown in boxes in the qualitative model, are the feed rate and the coarseness of the ore entering ball mill 1 (Cin and Vin), the rate of water addition into the two ball mills (Water1 and Water2), the power drawn by the two ball mills (PowerBM1 et PowerBM2), the coarseness of the ore at the screen (Cscreen), the coarseness and output rate of ore leaving the system (Cout and Vout) and the overall

efficiency (Efficiency). Given the values of these parameters at time T, the learning task is thus to predict whether the efficiency will have increased or decreased by time T+ΔT.

The grinding circuit simulator which was used to generate data sets was a simplified version of a more complex, commercial simulator used in the mining industry [JKTech Ltd., 1991].

3.1.2.3 The analog electric circuit

The analog electric circuit is shown in Figure 3.8 The boxes labeled SUM and INTF represent small elementary subcircuits constructed around an operational amplifier.

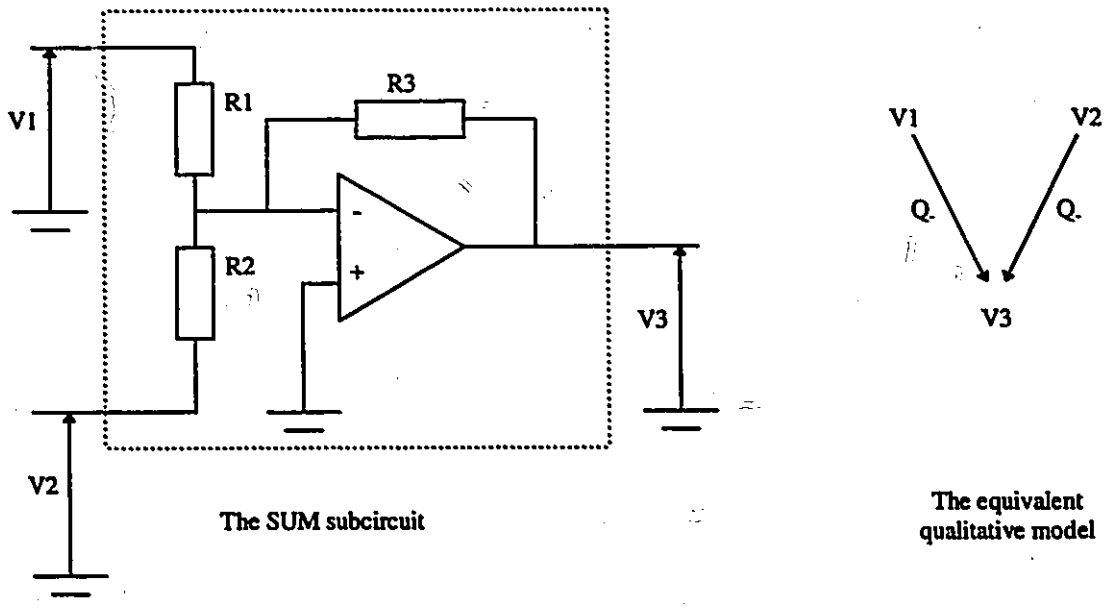


Figure 3.6 The SUM subcircuit

The SUM subcircuit's function is to add the two input voltages, then to invert and amplify this sum. Figure 3.6 shows the internal structure of the SUM circuit at the component level and its corresponding qualitative model. R1, R2 and R3 are resistors and are not parameters of the circuit in the sense that their values do not vary.

The INTF sub-circuit is a modified version of an inverting integrator in which a feedback loop was added. This sub-circuit thus initially behaves like an ordinary integrator (the output is the opposite of the integral of the input voltage) but, since a portion of the output is fed back into the input, it quickly converges to an equilibrium value. Figure 3.7 shows the internal structure of the INTF sub-circuit and its corresponding qualitative model.

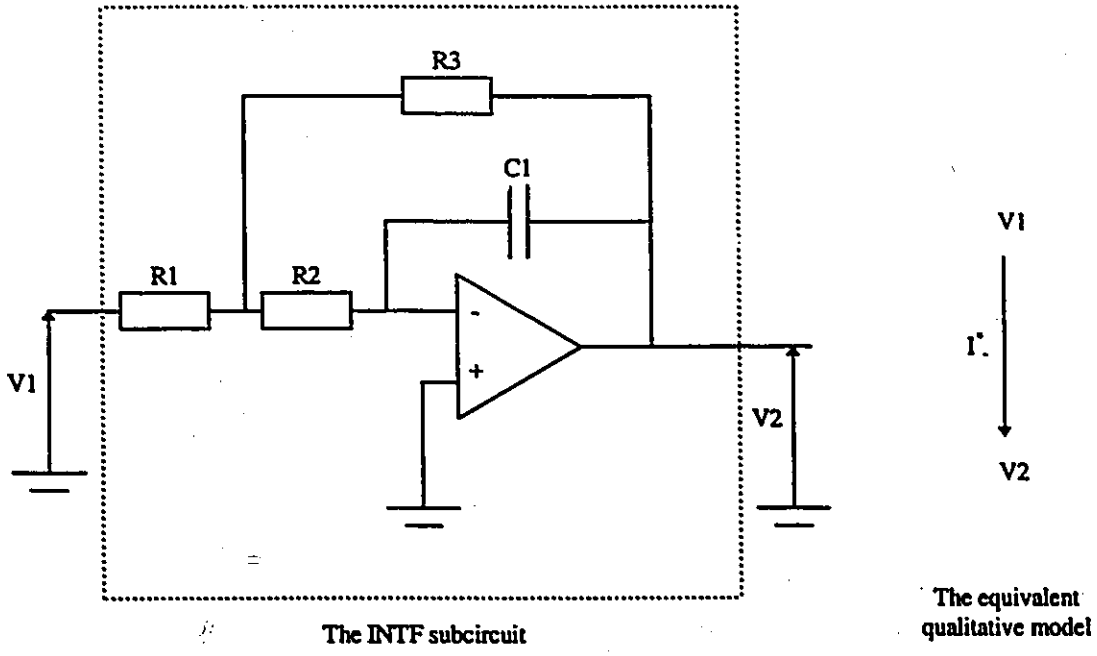
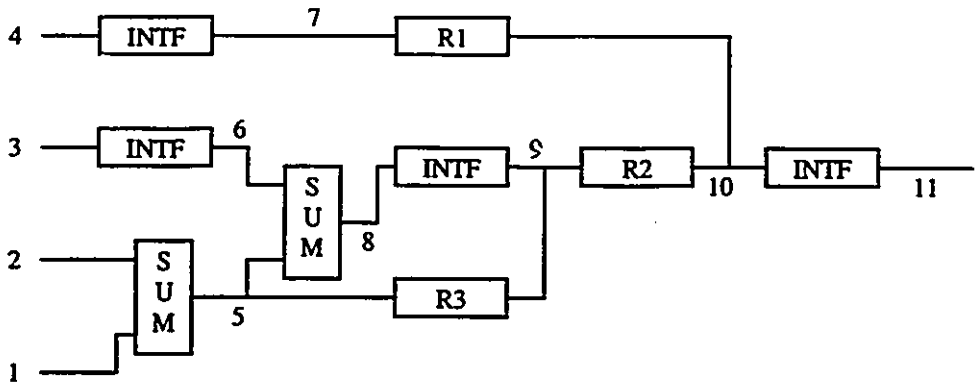


Figure 3.7 The INTF sub-circuit

In Figures above and below, components R1, R2 and R3 are simple resistors.

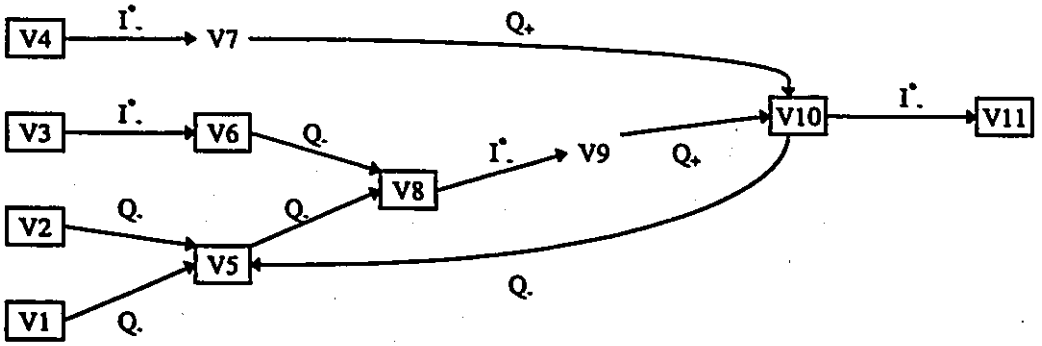
The qualitative model of an analog electric circuit is shown in Figure 3.9. Once the qualitative models for the SUM and INTF sub-circuits are defined, the building of a qualitative model for the whole circuit merely consisted in linking the small models.



The analog electric circuit

Figure 3.8

The operator controllable parameters are the four input voltages V1, V2, V3 and V4. The target parameter is the output voltage V11.



Qualitative model of the analog circuit

Figure 3.9

There are nine observable parameters, respectively: V1, V2, V3, V4, V5, V6, V8, V10, and V11. V7 and V9 are not observable. This choice was made to test the behaviour of our system on non-observable parameters. Given the value of the observable parameters at time T, the learning task is to predict whether the output voltage V11 will have increased or decreased by time T+ΔT.

To simulate this circuit, the HSPICE software package was used. HSPICE is a derivative of SPICE (Simulation Program with Integrated Circuit Emphasis) and is intended for use with UNIX workstations.

3.2 From the model to explainable rules

We will see later that our inductive tool induces, or learns, rules which try to achieve the best classification of a given set of examples. In this section, we will describe our definition of which rules are *explainable* with respect to a qualitative model, while noting that other alternatives might be also acceptable.

3.2.1 Rule structure

We define a rule as a structure:

if T_1 *and ... and* T_n *then C*

where T_i (for i between 1 and n) is a test on some observable parameter P_j, i.e. $P_j<k_i$ or $P_j>k_i$ where k_i is a real constant value, and where C asserts either "Pconc will increase" or "Pconc will decrease" during the next time-step. Pconc does not have to be observable and can be any parameter of the application domain.

We also define a **rule schema** as being a rule whose constant values are not specified. In other words, the rule schema corresponding to a given rule is the set of rules described when all the constants values of the parameter tests, the k_i for i between 1 and n, take all the possible values permitted for their associated parameter.

By definition, we note a rule schema as follows:

if P3>_ then P6 will increase

Thus, if the set of permitted values for P3 is {1,2,3,4,5}, then the rule schema is the set:

{ if P3>1 then P6 will increase
if P3>2 then P6 will increase
if P3>3 then P6 will increase
if P3>4 then P6 will increase
if P3>5 then P6 will increase }

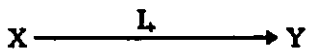
Note that, in our domain applications, all the parameter take their values in $\Re$, so the corresponding rule schemas contain an infinite number of rules. In the following, the word 'rule' may also refer to 'rule schema', for the sake of brevity.

3.2.2 Fully specified rules

3.2.2.1 Rule schemas corresponding to single links

The main goal of our modeling of physical systems as qualitative models is to allow us to have an automated way to distinguish between rules that are explainable with respect to our knowledge of the system and those which are not explainable. Thus, we must build a correspondence between a set of qualitative links forming a graph and the set of rules explainable according to these links. Let us consider, first, the case where there is only one qualitative link.

If we consider the link I+ in isolation, remembering that the meaning of:



is "if X is high, then Y will increase"[1], then we wish to consider the rule "if X>_ then Y will increase" as being explainable by the qualitative link above. We do not wish, however, to consider the rule "if X<_ the Y will increase" as being explainable. This is because "X is high" is the qualitative translation of the fact that X is greater than some value. Similar

[1] "X is high" means "X is greater than a given value", as introduced earlier (see page 20).

interpretations can be performed for each different type of link we use in our qualitative models, which lead to an associated rule schema (possibly empty) we consider explainable with respect to the qualitative link. Figure 3.10 shows the schema associated with each type of link.

Relations and their corresponding rule schemas

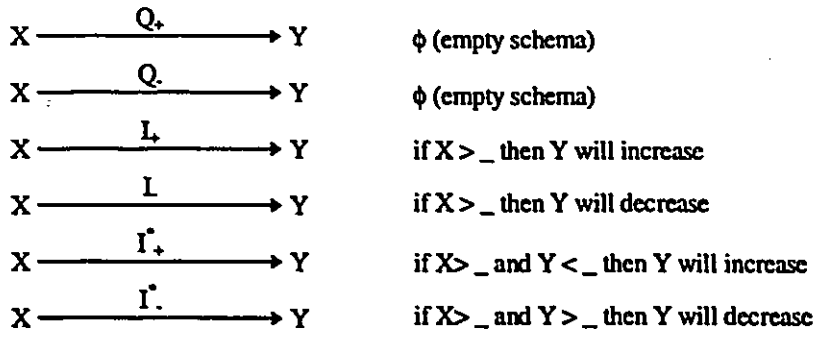


Figure 3.10

In the case of an I* relation, there exists an equilibrium value for Y corresponding to each value of X. If we consider the I*+ relation for example, we know that this equilibrium value for Y increases monotonically with the value of X (that is what the + means). Thus, if we know that X is above a given value and that Y is below the corresponding equilibrium value, then we can predict than Y will increase, until it eventually reaches this equilibrium value. This is thus different from the I link itself, for whom, the value of X alone determines whether Y will increase or decrease. Note finally that for the Q link, knowing the value of Y alone does not give us any information on how Y might change in the next time-step, so the corresponding rule schema is empty.

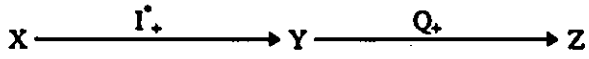
We can already notice that only I/I* relations have a prediction effect. It is because only these relations can make an affected parameter vary even when the 'input' parameter is stable. Q links just propagate variations of the system parameters, while I/I* links can create these variations. In this sense, I/I* links represent imbalance in the system: each one of the I/I* relations is capable of creating a variation of one of the system's parameters,

which will then propagate to the target parameter. Reciprocally, we want to consider as explainable rules that predict the variation of the target parameter, and for which this variation can be explained as the propagation of another parameter variation, itself created by one I/I* of our qualitative model, and those rules only.

By definition, for an I/I* link oriented from the parameter X to the parameter Y, we call X the **'cause'** and Y the **'pre-effect state'**. Rules derived from I* links must test the pre-effect state. Rules based on I links need not.

3.2.2.2 Propagation paths

The rule schemas of Figure 3.10 can be seen as the set of rules which we consider explainable with respect to a qualitative model containing only two parameters and only one link. We now generalise this notion to apply to any qualitative model. First, we note that we are only interested in rules which predict the evolution of the target parameter, for only these fit in our learning task. It does not mean that being able to predict the evolution of another parameter is useless: there might exist a path in the qualitative model from this parameter to the parameter of interest which would allow us to deduce how the latter will change. If such a path exists, then there exists a rule schema corresponding to the global path in the qualitative model (i.e. the I/I* link plus the path to the target). We call the path along which we are able to propagate the effect induced by the I/I* link to the target a **propagation path.** The following example shows a basic case in which we can derive a propagation path. If X is observable and if Z is the target parameter, and if we have the following links:

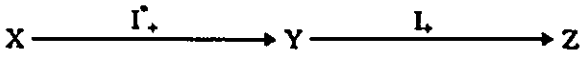


then the associated rule schema is

if X>_ and Y<_ then Z will increase

The qualitative explanation is simple: "if X is high and Y is low then Y will increase (because of the I*+ link connecting X and Y), and if Y increases then Z increases (because of the Q+ relation)". The I/I* link is responsible for Z's future change and the Q+ link acts as a propagator of this cause.

Q+ and Q- are obvious propagators. If we know how one of two parameters connected by a Q link will change, we may deduce the change of the other: if the link is Q+, they will vary the same way (they both increase or they both decrease) ; if the link is Q- then if the first increases, the second will decrease (similarly, if the first decreases, then the second parameter will increase). I/I* links can also act as propagators. In the following example,



high X makes Y increase ; furthermore we also consider that Y increasing makes Z rise too. There is a part of interpretation in this. The fact that Y is rising does not strictly tell us anything about its current value but we consider probable, thus explainable, that Y will reach a value big enough to make Z increase. Hence, for this example, the set of explainable rules would be the union of the rule schema associated with each link, i.e.

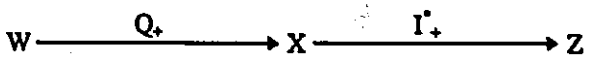
(if X>_ and Y<_ then Z will increase) ∪ (if Y>_ then Z will increase)

Of course, in the last example, if the second I+ had been oriented from Z to Y, then the set of explainable rules would have been empty: there is no possible propagation of the effect induced by X being high from Y to Z and the second I+ from Z to Y does not give us any information about Z's future change. In other words, when we try to find a propagation path from an I/I* link to the target, we are allowed to follow any link, as long as they are oriented from the pre-effect state to the target along the path. Note finally that the effect on the target parameter (either 'will increase' or 'will decrease') is inverted each time we go through a Q-/I-/I*- link: if we know that a parameter A will increase and if there is a I-

relation from A to B, then we predict that B will decrease, and so on, all along the propagation path.

3.2.2.3 Non observable parameters

When examining a particular I/I* link in the qualitative model to check whether there exists a propagation path from this link to the target, it might happen that the cause or the pre-effect state (see definition p. 34) or both are not observable. For an I link for example, if the cause is not observable, then we cannot apply the corresponding rule schema, which involves a test on the cause parameter. Similarly, for I* links, both parameters are tested upon. When this problem arises, we can resort to the same kind of reasoning we used to build propagation paths. In other words, we can try to find indirect evidence of the state of a non observable parameter through a path to another parameter, which is observable. A simple example of this is as follows: if we have the links

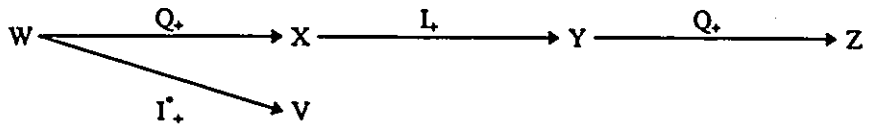


and if Z (the target parameter) and W are the only observable parameters, we will obtain the following explainable rule:

if W>_ and Z<_ then Z will increase

We in fact replaced the expected test on X by a test on W, which is equivalent since they are linked by a Q+ relation, i.e. they are "qualitatively proportional". The corresponding qualitative explanation of this rule schema is "if W is high then X is high (because of the Q+ relation linking them) and if X is high and Z is low then Z will increase (because of the I*+ link connecting X and Z)" or "If X is high, as evidenced by W being high, and Z is low then Z will increase".

We also allow paths used for indirect evidence to contain I* links. The reason for this is that, if the part of the system to which the I* belongs is stable, the link itself will behave like a Q+ link. For example, if the qualitative model contains the following links:



and if X and W are not observable but V is, then we want the rule

if V>_ then Z will increase

to be considered as being explainable. The corresponding qualitative explanation is "if W is high, as evidenced by V being high then X is high ; if X is high then Y will increase, thus Z will increase".

We do not consider I links when looking for indirect evidence of the state of a non observable parameter. The reason is that we look for a justification of something of the form "X is high" or "X is low". The only information we can derive from I links is about the change of a variable ("X will increase/decrease"), not its state. I* relations present the property of behaving like Q relations under some conditions, and this is why they are allowed in our evidence path, though I links are not.

To find evidence of the state of a non observable parameter C, we proceed as follows:

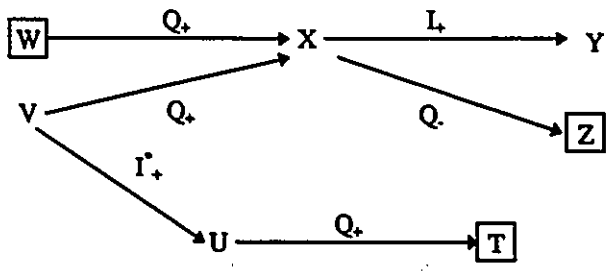
we find the shortest paths from C to all other nodes reachable by the following algorithm, i.e. reachable by either method a, b or c (remember that we do not allow I links in these paths, only Q and I* links):

a. go forward from C to some observable parameter. This means that we follow all the Q+ and I links, starting from C. It is an exhaustive breadth-first search and a branch of the search stops when it reaches an observable parameter or fails when it cannot go further.*

b. go backwards from C to some observable parameter.

c. go backwards from C to some non-observable parameter, then forward to some observable parameter.

We are interested in all the observable parameters extracted by the algorithm, not just one. In fact, the operation we perform on the original rule schema, in order to take into account that a cause or a pre-effect state is not observable, is equivalent to replacing the corresponding parameter test by the conjunction of tests on all the extracted observable parameters. Here is a simple example:



Y is the target parameter. T, W and Z are the only observables. We consider the I+ link from X to Y and we want to find indirect evidence of "X is high" to predict that Y will increase. We apply the algorithm and the result of the three searches are:

a. We follow Q and I* links starting from X. The only node reachable is Z, which is observable. Since X and Z are connected by a Q- relation, "X is high" corresponds to "Z is low"

b. We go backwards from X. There are two nodes we can reach: W and V, but V is not observable so it is useless at this point. W and X are connected by a Q+ relation, so "W is high" is possible evidence for "X is high".

c. We go backwards from X to some non-observable parameter, then forward to some observable parameter. First we reach V by going backwards from X. From V, we

can reach U and T by going forward, but only T is observable. On the path leading from X to T, there are only "+" relations (Q+, I*+, then Q+ again), so "X is high" is confirmed by "T is high".

Then, we collect all our evidences and replace the test corresponding to "X is high" in our rule schema (if X>_ then Y will increase) by the conjunction of all the tests on our evidence parameter. The final rule schema is:

if T>_ and W>_ and Z<_ then Y will increase

3.2.2.4 Fully specified rules: definition and extraction algorithm

We have seen that explainable rule schemas are tightly related to I/I* relations. If there are no such links in a qualitative model, then we cannot derive any explainable rule from it. This is due to the fact that the information about the physical system provided to the inductive tool gives the values of each parameter at a given time. This constraint is imposed in order to be able to learn rules of the granularity adequate in controlling and describing physical processes. Thus, in order to derive a conclusion a parameter's future change, we need at least an I/I* link. This would not be true, if the changes of the parameters, rather than their values, were provided to the inductive tool.

Furthermore, a rule schema can be associated with each I/I* link, if we can find a propagation path from the cause to the target. Note that we also have to find indirect evidence when the cause or the pre-effect parameters (or both) are not observable. If we cannot find indirect evidence for a non-observable cause, then there is no rule schema corresponding to the considered I/I* link. For an I* link, the associated rule schema also tests the pre-effect state, so indirect evidence has to be found when the parameter involved is not observable. However, we are less restrictive for the pre-effect state than for the cause: if the cause is observable but the pre-effect state is not, we keep the rule schema with the test on the pre-effect state removed. The reason why we are less restrictive for the pre-effect parameter is that we consider it less necessary than the cause. If we do not

know that the condition on the cause is verified, then we do not consider that the rule can be used. If the pre-effect state cannot be known and if we know that the condition on the cause is satisfied, then we allow the rule to fire.

By definition, we call a **fully specified rule** (schema), a rule schema based on a I or I* link for which there exists an propagation path from the pre-effect state to the target and for which tests on non observable parameters have been replaced by tests on indirect evidence parameters. If the cause parameter is not observable, then indirect evidence of its state through other observable parameters must be found or the corresponding fully specified rule is not valid. If the pre-effect state is not observable, we find indirect evidence, if possible. If not, we just remove the test on the pre-effect state.

Note that there may exist different propagation paths in the qualitative model from the pre-effect state parameter to the target. It may also happen that, depending on the path we consider, we will produce different predictions concerning the target, i.e. if X is the cause of the I/I* link and Z the target, there might exist a path which corresponds to "X is high then Z will increase" and another one saying "if X is high then Z will decrease". In fact, we consider only the shortest path from the pre-effect state to the target. The other paths are ignored. Thus we will derive at most one explainable rule schema from each I/I* link.

Below we give the algorithm used to extract fully specified rules from a qualitative model. The algorithm only extracts rules that predict that the target will increase. In fact, the ones predicting the fall of the target are the same, with only the parameter tests inverted (X>_ becomes X<_). Thus, we need only to keep the ones predicting a target rise in storage.

The algorithm is:

1. Find any I/I connecting a node C to a node E*

2. Find the shortest path from E to the target T

3. C may not be directly observable. If this is the case, find the shortest paths to observable parameters reachable by the algorithm given in section 3.2.2.3. If no observable parameters are found then give up the corresponding I/I link and start again with the next one.*

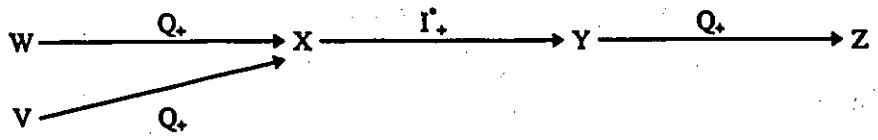
4. Do the same to find observable evidence of E's value if the current link is I and E is not itself observable. Note that we continue even if no evidence is found at this point.*

5. Store the corresponding fully specified rule which predicts that the target will increase (either this one or the inverted one)

The reason we keep rules for which no evidence of the pre-effect state was found is that, as explained later, we also consider schemas testing only a subset of the parameters of a fully specified rule as being explainable. We only enforce that there remains at least a test on the cause. Thus, rule based on I* links not testing the pre-effect state will still be considered as explainable.

3.2.3 Explainable rules: definition

In fact, we do not consider only fully specified rules to be explainable. We also accept rules that test only a subset of a fully specified rule test set, as long as this subset contains at least one test on the cause parameter (or on an observable parameter used as indirect evidence of the cause parameter). For example, if X is not observable but all the other parameters are, and if Z is the target parameter, there are 6 rule schemas corresponding to the links below (we consider only the rules which predict that the target will increase).



Qualitative Model (above)

Fully specified rule:

if V>_ and W>_ and Y<_ then Z will increase

Explainable rule schemas:

if W>_ and Y<_ then Z will increase

if V>_ and Y<_ then Z will increase

if V>_ and W>_ and Y<_ then Z will increase

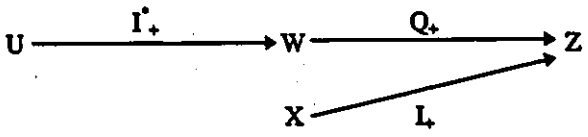
if W>_ then Z will increase

if V>_ then Z will increase

if V>_ and W>_ then Z will increase

Our derivation of fully specified rules assumed that only one source (I/I* link) was affecting the target during a time-step. Now we wish to consider rule schema corresponding to combinations of two effects on the target. The way to obtain these rule schemas is to combine all the subsets of attribute tests of each pair of fully specified rules. Here too, we require that the resulting rule must contain at least one test on the cause for each of the two original fully specified rules. The example below shows the rule schemas we obtain by combining two fully specified rules (i.e. the rule schemas derived from each one and the rule schemas derived from the combination of the two).

Qualitative model:



Fully specified rules:

1: if U>_ and W<_ then Z will increase (this fully specified rule is based on the I*+ link connecting U and W)

2: if X>_ then Z will increase (this fully specified rule is based on the I+ link connecting X and Z)

Explainable rule schemas:

if U>_ and W<_ then Z will increase	(1)
if U>_ then Z will increase	(1)
if X>_ then Z will increase	(2)
if U>_ and W<_ and X>_ then Z will increase	(1 & 2)
if U>_ and X>_ then Z will increase	(1 & 2)

However, we have to check that there are no conflicts created by the combination of two fully specified rules. There are two types of conflict we want to avoid, conflicts on the propagation path and conflicts on the indirect evidence path. The first ones occur when the two fully specified rules predict two different changes (such as Y will increase and Y will decrease) for one common variable they both have on their own propagation path. In this case, we consider that these two rules are incompatible (they predict two different behaviors for the same variable) and cannot be combined.

The second type of conflict occurs when the two fully specified rules assume a different state for one common parameter which belongs to their evidence path (for the cause as well as for the pre-effect state). For example, rule 1 may say "observable A is high then X is high then C1 is high" while the second says "observable B is high, so X is low, so C2 is low". Again, if this type of conflict occurs, we call the two rules incompatible (they assume two different states of the same parameter in order to be fired) and we do not combine them.

Note however that the second type of conflict does not always result in inconsistent rules. The corresponding tests are X>_ and X<_ which can be both satisfied by X>20 and X<30 for example. However, to keep things comprehensible, we enforce this extra constraint. Furthermore, the enforcement of these two constraints requires that the propagation path as well as the indirect evidence paths are stored with the fully specified rules when they are extracted.

In theory, we could similarly combine 3, 4 , or any finite number of fully specified rules. However, we chose not to go beyond two rules for two reasons. The first one is that it introduces a fair bit of complication in the algorithm: when trying to combine n rules, one has to check that they are not incompatible, which is not *very* hard to implement but which is not worth the effort since in practice, rules induced by CN2 on our application domains are short. The probability that a given rule would actually correspond to the combination of more than two fully specified rules (i.e. at least three parameter tests) is thus quite low. The second reason, and the most important, is related to our notion of explainability. We consider that the overall interpretability of an explainable rule decreases with the number of fully specified rules which were combined to form it. If we allow combination of too many fully specified rules, then we lose an essential property of the qualitative model which is to actually describe the influence of each parameter alone on the target. The rules we would induce would be much less interpretable, in the sense that we would allow influences of more than two cause parameters to be combined. Instead, we want shorter and clearer rules. We hope that very complicated rules can be decomposed in a set of shorter ones, which allow no more than two influences at a time. In fact, it is equivalent to say that we assume that, for each state of the physical system, the future change of the target parameter in the next time-step can be explained by the values of at most two parameters in the system.

We can now formulate our definition of which rules are explainable with respect to a given qualitative model. Let us call T(R) the set of attribute tests of a rule schema R. We say that a subset of T(R), where R is a fully specified rule, is a **proper subset** if it contains at

least one test on the cause (i.e. a test on the cause parameter itself if it is observable, or a test on another observable parameter used as an indirect evidence).

Definition: A **rule schema R** is ***explainable*** with respect to a qualitative model if and only if there exist **two fully specified rules R1 and R2**, extracted from the qualitative model as explained earlier, such that **T(R)** is either:

- a proper subset of **T(R1)**

- a proper subset of **T(R2)**

- included in the union of two proper subsets of **T(R1)** and **T(R2)** with **R1** and **R2** compatible (in the sense that they do not create conflict when combined).

Thus, a rule (not a rule schema) is explainable if it is included in an explainable rule schema, and non explainable otherwise. By extension we will call explainability of a ruleset with respect to a qualitative model, the percentage of rules of this ruleset which are explainable. For example, if a ruleset contains 6 rules and if only 4 out of these are explainable, then the explainability of the ruleset is, by definition, 4/6 = 0.6666... thus about 66.67%

3.2.4 Compiling explainable rules into a static structure

Now that the complete definition of the explainability of a rule has been given, let us consider how it will be taken into account, from an implementation point of view. As it will be explained with more details in the next chapter, our inductive tool (derived from CN2) performs a general to specific beam-search for the rules that best classify a set of training examples. It searches in turn for rules predicting a rise of the target parameter and for rules predicting its fall. For each kind, the search starts with the most general rule "if true then target will increase (respectively decrease)", and specialises it by adding some tests on observable parameters, one at a time. We use a heuristic which allows us to select

the best specialised rules, then these ones are specialised in turn, until the maximum search depth allowed is reached. Then the best of all the rules generated by the process (i.e. the best of the rules with 1, 2, ..., MaxDepth tests) is added to the produced ruleset.

What is important to us is to be able to control at any state of the search if a candidate rule is explainable, and even, if wanted, to constrain the search to only those rules which are explainable. Rather than looking back into the qualitative model and extracting the fully specified rules for each rule we examine, we store the fully specialised rules once and for all in a file. Also, we do not want to have to combine all the possible pairs of fully specified rules to determine whether a rule is or is not explainable. The solution which was adopted is to preprocess the model and store all the explainable rule schemas with the fully specified rules in a file, which will be loaded later by the induction module. This choice was guided by efficiency reasons.

To improve the performance in the case when we want the search to be constrained to only explainable rules, the explainable rule schemas are stored in a specialisation lattice form. The advantage of the lattice form is that, for each rule which is explainable and which we want to specialise, the corresponding entry in the lattice will give the list of all of the possible explainable specialisations of the rule. Since the lattice is implemented as a list of assertions of the form "current state" -> "list of explainable specialisations", a hash index was introduced to reduce the look-up time as much as possible.

Note that the lattice is generated once and for all, and unless the qualitative model is modified, it does not have to be updated. Note also that in each entry of the lattice there is a reference to the fully specified rules which explain the current state of the candidate rule and the further specialisations. This will be used to automatically generate qualitative explanations of the (explainable) rules of the ruleset and this is also why the fully specified rules were stored with the lattice. See Appendix A for an example of a lattice file.

Chapter 4

Induction of Explainable Concepts

In this chapter, we first describe our inductive tool. Then, we present our modified algorithm which is intended to learn explainable rules only. We will see how the data structure derived from the qualitative model and representing all the combined fully specified rules in the model, is integrated in this learning process. We end the chapter with a discussion about the limitations of this first and simple approach.

4.1 Overview of CN2

The inductive algorithm we use is the latest version of CN2 [Clark and Boswell, 1991]. It induces an unordered list of classification rules from examples. It uses Laplace's error estimate as its search heuristic, where for example, C4.5 uses an entropy-based measure.

4.1.1 The CN2 algorithm

CN2 is an algorithm designed to induce 'if... then...' rules in domains where there might be noise. The original algorithm, described in [Clark and Niblett, 1989] and [Clark and Niblett, 1987], was using entropy as its search heuristic and was only able to generate an ordered list of rules. As shown in [Clark and Boswell, 1991], using Laplace error estimate as a heuristic significantly improves the algorithm's performance. Furthermore, while ordered lists of rules present the advantage that they do not need any machinery for resolving clashes between rules (the way they are constructed ensures that only one rule can fire at a time), there is also a corresponding problem when the ordered set of rules

grows in size: since any single rule is dependent on all the others that precede it in the rule list, the further down in the list a rule is, the more difficult it becomes for a reader to understand its true meaning. This consideration led to the modification of the algorithm so that it can generate a set of non ordered rules, i.e. a set of independent rules.

Below is the CN2 algorithm, applied to our learning task: the attribute test are tests on observable parameters and there are only two classes, increase and decrease.

```
procedure CN2(allexamples)
        let ruleset = { }
        for each class in {increase,decrease}:
                generate rules by CN2ForOneClass(allexamples,class)
                add rules to ruleset
        return ruleset

procedure CN2ForOneClass(examples,class)
        let rules = { }
        repeat
                call FindBestCondition(examples,class) to find bestcond
                if bestcond is not null then
                        add the rule 'if bestcond then predict class' to rules
                        remove from examples all examples in class covered by bestcond
        until bestcond is null or examples is empty
        return rules

procedure FindBestCondition(examples, class)
        let mgc = the most general condition ('true')
        let beam = {mgc}
        let newbeam = { }
        let bestcond = null
        let depth = 0
        while beam is not empty and depth < maxdepth
                add all the specialisations of each cond in beam (i.e. generate all the cond' =
'cond & test' where test is a test on an observable parameter) to newbeam
                remove duplicates from newbeam
                sort newbeam according to Laplace expected accuracy
                let newbest = best condition in newbeam
                if newbest is better than bestcond then let bestcond = newbest
                while size of newbeam > beamwidth then remove the worst
```

```
condition in newbeam
            let beam = newbeam
            let depth = depth + 1
      return bestcond
```

There are two user-defined constants maxdepth and beamwidth which are respectively the maximal length of the induced rules (the maximum number of tests in each rule) and the maximum size of the beam (how many candidate rules we keep to be specialised further).

When generating a specialisation, we check that the resulting coverage is indeed improved. If a specialisation has the same coverage as the original condition, or even a worse one (less positive examples are covered and still the same number of negative examples are covered), we reject it. Thus, it can happen that the search stops before all positives examples are covered for a class. CN2 is not intended to look for a perfect coverage of the training examples set but rather to induce rules which perform well on unseen examples in noisy domains.

Furthermore, since the obtained ruleset is not ordered, a clash between rules can happen, i.e. it might be possible that more than one rule can fire to predict the class of an example. We resolve these conflicts by tagging the rules with the number of examples it covers of each class. If a clash occurs, we total the distributions of the examples covered by all the rules that fire and the predicted class is the one that has the highest value. For example, if R1, R2 and R3 cover an example whose class we want to predict and if the distributions of covered examples for each rule are:

	increase	decrease
R1	13	0
R2	2	10
R3	20	0

then the totals are 35 and 10 for the class increase and decrease respectively. The predicted class is thus increase. An advantage of the way we resolve clashes between rules is that we take each rule into account, with an importance related to its example coverage.

This way, if we have 10 rules covering only 1 example of the class decrease and if we have a rule covering 100 examples of the class increase, we will predict the class increase. If we only counted the number of rules predicting each class, we would predict the class decrease and ignore the overwhelming coverage of the last rule, coverage which indicates a probably more accurate predictive power on unseen examples.

4.1.2 Laplace accuracy

When trying to learn the best rule which covers a set of examples, CN2 employs an evaluation function based on the Laplace error estimate. This function measures the expected accuracy of the rule we want to evaluate. In other words, it gives a value which reflects, according to the distribution of examples of the training set among classes, how well the rule will classify unseen examples. This expected accuracy, namely 1 - the Laplace expected error estimate, is given by the formula:

$$LaplaceAccuracy = (Nc + 1) / (Ntot + k)$$

where:

k is the number of classes in the domain
Nc is the number of examples in the predicted class c covered by the rule
Ntot is the total of examples covered by the rule

This formula is a special case of the m-probability-estimate developed by Cestnick [Cestnick, 1990] and analysed further in [Cesnik and Bratko, 1991]. In our domain, the number of classes k is equal to 2 (increase and decrease), and if we name Np the number of correctly covered examples from the training set and Nn the number of incorrectly covered examples, the formula becomes:

$$LaplaceAccuracy = (Np + 1) / (Np + Nn + 2)$$

It has been shown that the use of this metric as CN2's evaluation function improves significantly the results of the algorithm over the previously used entropy-based metric. It also avoids the undesirable 'downward bias' of the latter, which tends to make it select highly specific rules.

Another useful property of the Laplace expected accuracy is that it takes values bounded between 0 and 1. The worst value is 0 (we expect that none of predictions of the rule will be correct) and the best is 1 (for an ideally perfect rule. Note however that the value 1 is never actually reached). We will exploit this aspect later in chapter 5 by imposing a minimum expected accuracy during induction (See section 5.1).

4.2 Totally constrained search

In this section, we present our first approach of the learning of explainable rules in the presence of a qualitative model. It is also the simplest. In this approach, we restrict the search to the space of rules which are explainable with respect to the qualitative model.

4.2.1 Description

The modifications to the original CN2 algorithm to make it learn explainable rules only are straightforward once the lattice file is generated for the qualitative model (see chapter 3). In fact, during the specialisation operation of the candidate conditions currently in the beam, we only have to generate the specialisations listed in the lattice for each entry corresponding to one of the conditions (and those only). For example, if the beam contains the conditions:

p1>250 and p2<125
p3>470

We look up in the lattice at the entries p1> p2< and p3> for the corresponding class (either increase or decrease) to find out which are the tests we can add to our conditions, such that the rules resulting from the specialisation operation will also be explainable. Once the list of tests for each condition is known, we complete each of these tests with the best cut-off value according to the training examples (i.e. p4< becomes p4<345 where 345 is the best cut-off value). The best cut-off values are chosen according to the training examples covered by the condition to be specialised, and are intended to separate in the

best way the positive from the negative examples. For example, if we want to find the cut-off value k for the test p3 > k and the class increase, and if the covered examples are (we show only the values for the parameter p3):

decrease: 13, 24, 25, 29
increase: 31, 44, 179

then, the cut-off value will be 30 (average of 29 and 31) and the specialising test will be p3>30.

Because we exclusively use the lattice as a mean of generating specialisations, we ensure that we only construct rules which are explainable with respect to the qualitative model. Reciprocally, every explainable rule can be generated by the algorithm, if adequate training examples are furnished. We highlight this property because it makes the learning system consistent with our definition of explainability of a rule, as presented in the previous chapter.

There are two main objectives of this approach. The first one is to induce totally explainable rules. As already stated, such an explainability is essential for integrating the results of learning back into a knowledge base or for any practical application in general. The second one is to prune out rules which, by chance, perform well on the training data and which are not explainable. We expect these rules to have a poor predictive accuracy on unseen data and thus, we expect to improve the learning.

4.2.2 Interpretability of the induced rules

Predictive accuracy is the usual measure of an algorithm's performances. However, practical experience tends to show that the interpretability of the results is also very important. There are even some domains where predictive accuracy is hardly of interest to the user. Instead, what is important is how he can interpret the induced descriptions.

For example, in KARDIO [Bratko, Mozetic and Lavrac, 1989], inductive learning is aimed to generate a compressed and more interpretable version of their rules for prediction of heart disorders. The obtained concept descriptions are not intended to be used with unseen data since the training set is already a consistent and complete domain theory. Instead, readability and interpretability was what motivated the use of induction. In [Marchand, 1995], the interpretability of the decision rules produced by learning is presented as a key factor, determining how efficiently these rules will be applied in practice. If the rules have a low interpretability, then, as a consequence, errors in application of this rules by human users will appear more often.

In our first approach, the induced rules have a maximum interpretability, since they are explainable, with respect to some background knowledge represented by the qualitative model. In fact, a qualitative explanation is indeed built for each induced rule. These explanations are based on the fully specified rules extracted from the qualitative model (see chapter 3) and describe how the measured state of a parameter affects the future behavior of the target parameter. The example below shows a rule as in its operational form, followed by the corresponding qualitative explanation. It is very easy to understand the explanation just by following the corresponding links in the qualitative model (see Figure 4.1). In order to make the qualitative explanations of the rules shorter, we use the terms "high" and "low" to refer to "greater than a given value" and "lower than a given value", thus "V3 is high" should not be understood as the value of V3 is high in V3's range of possible values. Note finally that the list of rule/explanation pairs is the actual output of the algorithm.

In this example (see Figure 4.1), the predicted behavior of the target parameter V11 is the combinations of two distinct influences. The first one is the propagation of the fact that V3 is high through the path [V3, V6, V8, V9, V10, V11] and corresponds to the test V3 > -550 in the rule. The second one is the consequence of V10 is low through the I*-relation linking V10 and V11 and corresponds to the tests V10 < -40.0 and V11 < 135.5 in the rule.

RULE::

IF V10 < -40.0

AND V11 < 135.5

AND V3 > -550.0

THEN V11 will increase.

EXPLANATION FROM THE QUALITATIVE MODEL:

GIVEN

the voltage at node 3 is high

THEREFORE

the voltage at node 11 will rise.

(the voltage at node 3 is high,

therefore the voltage at node 6 will fall

therefore the voltage at node 8 will rise

therefore the voltage at node 9 will fall

therefore the voltage at node 10 will fall

therefore the voltage at node 11 will rise).

AND

GIVEN

the voltage at node 10 is low

BUT

the voltage at node 11 is low

THEREFORE

the voltage at node 11 will rise.

(the voltage at node 10 is low,

therefore the voltage at node 11 will rise).

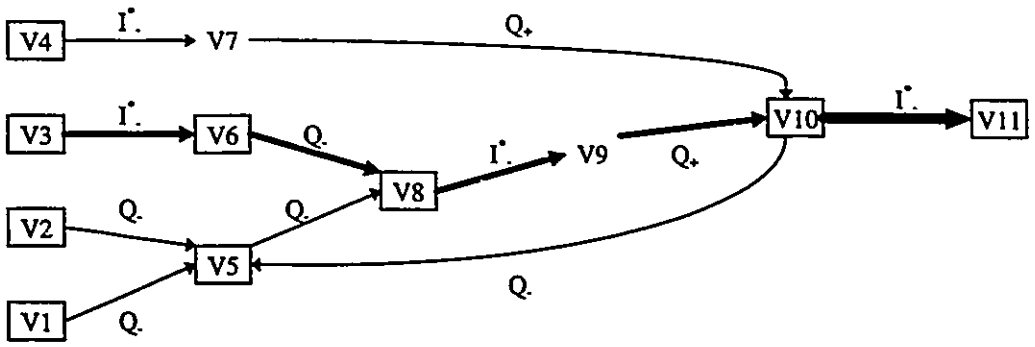


Figure 4.1 Qualitative model of the analog electric circuit

4.2.3 Limitations

This approach (totally constrained search) has two major limitations, both related to our background knowledge representation. First, it assumes a reasonably complete and correct qualitative model describing the physical system of the application domain. While it is not a very strong assumption for simple applications, the existence of such a model for more complicated systems can be more questionable. It is very easy to show that the predictive accuracy of the induced ruleset depends heavily on the quality of the qualitative model, with respect to the actual physical system. If the model is incomplete (some essential links are missing) or incorrect (there are some wrong relations in the model), then the induced rules will perform poorly on unseen data. Achieving total explainability for a ruleset which does not perform better than the default rule (predict the most common class in the training set) cannot be really claimed as a success. Even if, fortunately, a qualitative model has to be very deficient to provoke such catastrophic results, it illustrates the necessity of a refinement of the approach.

The problem arises when the knowledge injected into the system through the qualitative model is contradicted by the training examples. We reject rules which perform well on the training data and which are not explainable, even in the face of an overwhelming statistical

evidence. In the next chapter, we will present an approach which addresses this particular problem. We called this approach the threshold algorithm.

The second limitation is related more closely to the language we use to build our qualitative model. It happens often, when constructing a model, that one hesitates to insert a particular link. Even with a perfect knowledge of the actual relationships between variables, one may have to choose between one or more possibilities, which may not be equivalent. The relation between variables may not be monotonic but may depend on the state of some other parameters of the system. With the knowledge representation described in chapter 3, the expert has only two solutions, when hesitating about a particular link:

- he includes the link in the model and thus he agrees the fact that all the rules based on this link will be 'totally' explainable.

- he does not include the link and all the rules which could have been explained by this link will be rejected.

One way to solve the conflict is to allow the system to handle uncertainty. In the second section of chapter 5, we describe a third approach which uses belief weights in the qualitative model.

Chapter 5

Toward a trade-off between accuracy and explainability

This chapter describes two methods providing an alternative to the totally constrained search presented earlier. The first method is intended to deal with strongly incomplete or incorrect models by allowing the system to induce non explainable rules supported by strong statistical evidence in the data. We will also see how this method can be used to suggest revisions of the qualitative model. The second method allows some uncertainty when designing a model by using belief weights.

5.1 Compensating for loss of accuracy caused by the constraint of total explainability

5.1.1 Motivation

The method presented in chapter 4 learns only rules which are explainable with respect to the qualitative model. The technique used is to constrain the search to the space of rules which are explainable. Rules which are not explainable are never considered nor evaluated. This is not a problem when the background knowledge provided to the system, by the means of the qualitative model, is consistent and relatively complete with respect to the actual (or simulated) physical system.

But in fact, our qualitative model may not be perfect. If the model is incomplete, in the sense that some essential aspects of the system are missing in our description, then the space of explainable rules will not contain the rules corresponding to this missing

knowledge. As a result, some examples will be classified wrongly and the overall predictive accuracy of the rules induced by our system will be poor, as opposed to the accuracy obtained with the standard inductive algorithm, using no background knowledge. The same loss of performance is also expected when the model is incorrect, i.e. it contains relations which are not consistent with the behavior of the physical system itself.

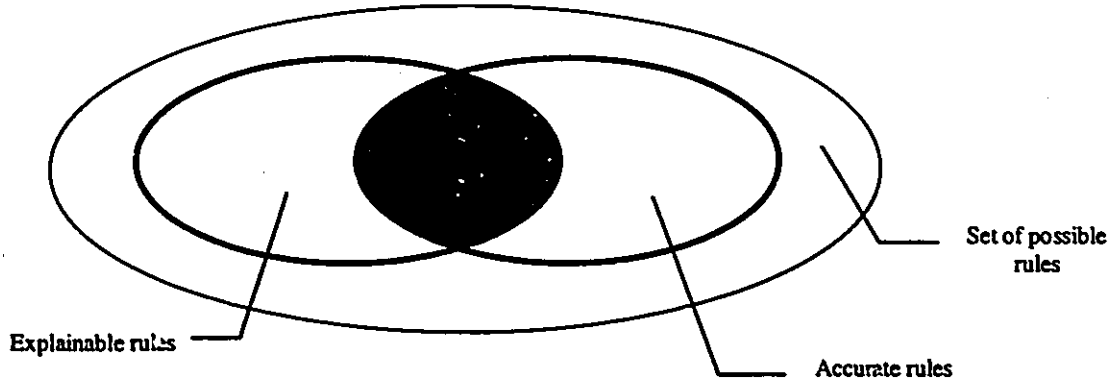


Figure 5.1 Intuitive view of the search spaces

Figure 5.1 gives a more intuitive perception of the problem. The standard inductive algorithm heuristically searches in the space of rules for rules which have a high predictive accuracy. We thus expect to induce rules which belong to the set of accurate rules. Our totally constrained search described in chapter 4 looks for rules, which also classify well the training examples, but in the subset of rules which are explainable with respect to the qualitative model. The resulting rules will probably belong to the intersection of the two subsets, i.e. the darkened area. We can then distinguish two extreme situations. On the one hand, if the model is good, then the overlap between the set of explainable rules and the set of rules which classify well the data is important and the induced ruleset is both explainable and accurate. On the other hand, if our model is very deficient (incomplete, incorrect or both), the overlap of the two sets will be small or empty. In this case, if we induce rules with the standard CN2 algorithm, we will still obtain rules with a good accuracy but which cannot be explained by the model. If we resort to the totally

constrained search, the induced ruleset will be totally explainable but will have a poor predictive accuracy. Neither alternative is really satisfactory.

The main problem we want to avoid is to achieve explainability of the induced ruleset by sacrificing the predictive accuracy. Our application domains are all related to process control and for these, predictive accuracy is also very important if we want the system to behave correctly when guided in its decision by the learned ruleset. On the other hand, as already discussed in chapter 4, the interpretability of the results of the learning is essential too. We thus see the necessity of a trade-off between these two goals. The method we present in this section is intended to learn ruleset with high explainability (not necessarily perfect) but also high accuracy. To achieve this, we will modify the totally constrained search to allow non explainable rules with high predictive accuracy to be induced under some conditions.

5.1.2 Description

While we want rules with a high predictive power, we do not want to use the standard CN2 algorithm because it produces rules which are very poorly explainable. We will see later in chapter 6 that, for example, for the water pipes domain, about only 50% percent of the rules learned by CN2 are explainable with respect to our qualitative model. It tends to show that, if we want highly explainable rules, we have to bias the search to favour those rules. Our first method, the totally constrained search, has the strongest bias possible since it allows only explainable rules to be considered. The disadvantage of this method appears when the model is poor. As a consequence, induced rules have a poor predictive power. In this method, we try to detect when this bias has some strong negative results on the predictive power of the induced rules.

The solution we adopted is to constrain the search to explainable rules (and thus to bias the search to favor explainability) unless there is some evidence that the induced rules perform poorly on the training data. Once such a situation has been detected, we change

our search strategy and concentrate on the accuracy of the rules we induce. Thus, we may at this point consider rules which are not explainable with respect to the qualitative model and if they perform better than the explainable ones, then they will be selected. In other words, we induce explainable rules unless there is some statistical evidence that they do not have a predictive power as high as expected. If this happens, we just choose the best rules, no matter if they are explainable or not.

The difficulty in this approach is to evaluate in an objective manner whether a given rule performs poorly. Obviously, this evaluation has to be done only on training data. One solution is to rely on Laplace expected accuracy. As already explained in section 4.1.2, Laplace expected accuracy estimates what the predictive accuracy of a candidate rule will be on unseen data. The values given by this measure are bounded between 0 and 1, so it is very easy to impose a minimum expected accuracy on the candidate rules during search

Our method is more precisely described in the algorithm below. It is very similar to our totally constrained search presented earlier. The only modified procedure is the procedure FindBestCondition. Given the set of non covered yet positive examples, it returns the rule which best classifies these positive examples. It starts with the most general rule and specialises it. Then the best specialisations in the current beam are kept and specialised further. With the totally constrained search, only explainable specialisations, i.e. those included in the lattice file, were considered. In this version of the procedure, we first specialise by using the lattice. Then we check that at least one of the specialisations in the current beam has an expected accuracy greater than a given user-defined threshold.

If at least one[1] specialisation has an expected accuracy above the threshold, then we keep all the explainable specialisations in the beam. Thus, if at every depth, at least one of the explainable specialisations passes the threshold test, the search will produce the same results as the totally constrained search.

[1] Remember that this is an hill-climbing search with a limited beamwidth. At the end, only the best rule will be added to the set of learned rules. Thus, as long as there exists one explainable rule whose expected results are satisfactory, there is no need to consider non-explainable rules.

If, at a given depth, none of the explainable specialisations in the current beam has an expected accuracy higher than the threshold, then we give up the current specialisations and backtrack to the previous depth. Further operations are not constrained by the lattice, i.e. we try all the specialisations, no matter if they are explainable or not. At this point, the algorithm indeed behaves like the standard CN2 algorithm, and does not take the background knowledge into account until the end of the execution of the procedure.

Similarly to CN2 (and also to the totally constrained search), the threshold algorithm executes an hill-climbing search, keeping the best candidate rules in the beam to be specialised further. The difference with this approach is that its bias for explainable rules can be given up during search, according to some evidence found in the data that explainable rules will perform poorly (that their predictive accuracy will be below a given value).

Below is the modified procedure FindBestCondition. All the other procedures of the algorithm are identical to those presented in section 4.1.1.

```
procedure FindBestCondition(examples, class)
        let mgc = the most general condition ('true')
        let beam = {mgc}
        let bestcond = null
        let depth = 0
        let use_lattice = true
        while beam is not empty and depth < maxdepth
                let newbeam = { }
                if use_lattice then generate the explainable specialisations (as indicated by
the lattice) of the conditions in beam. Otherwise, generate all the possible generalisations.
                add all the specialisations to newbeam
                remove duplicates from newbeam
```

```
        sort newbeam according to Laplace expected accuracy
        let newbest = best condition in newbeam
        if use_lattice and
        ( if Laplace expected accuracy of newbest < threshold or if newbeam is
empty )
                then use_lattice = false
        Otherwise
                if newbest is better than bestcond then let bestcond = newbest
                while size of newbeam > beamwidth then remove the worst
condition in newbeam
                let beam = newbeam
                let depth = depth + 1
    return bestcond
```

Giving up the explainability requirement is actually an expansion of the search space from the space of explainable rules to the space of all possible rules. We allow this search expansion to take place when explainable rules seem to perform poorly, because it could mean that the qualitative model do not explain well the set of uncovered positive examples. In this case, we decide to rely on pure induction to learn the corresponding rule.

There is also a second situation in which the search expansion takes place. A quick look at the procedure above indicates that we also backtrack and expand the search when the set of specialisations proposed by the lattice is empty. The reason for this is that this situation could be caused by an incompleteness of the qualitative model. Thus we allow the search to expand to the set of all possible rules to check whether there exist some non explainable specialisations of the candidate rules in the beam with a higher expected accuracy.

Figure 5.2 shows a run of the procedure *FindBestCondition*. It was obtained for the water pipes domain and with a qualitative model where the I*+ link connecting L6 and L7 was

removed. The threshold value was 0.8, the maximum search depth allowed was equal to 4 and the beam width was equal to 5 and the class to predict is *increase*.

The first tree shows the state of the beam after one specialisation. Note that these specialisations are explainable, since we start by constraining the search to explainable rules only. Since the best specialisation has only an expected accuracy of 33.2 % (less than 80%, which is the threshold value), the search is expanded by allowing purely induced and non-explainable rules.

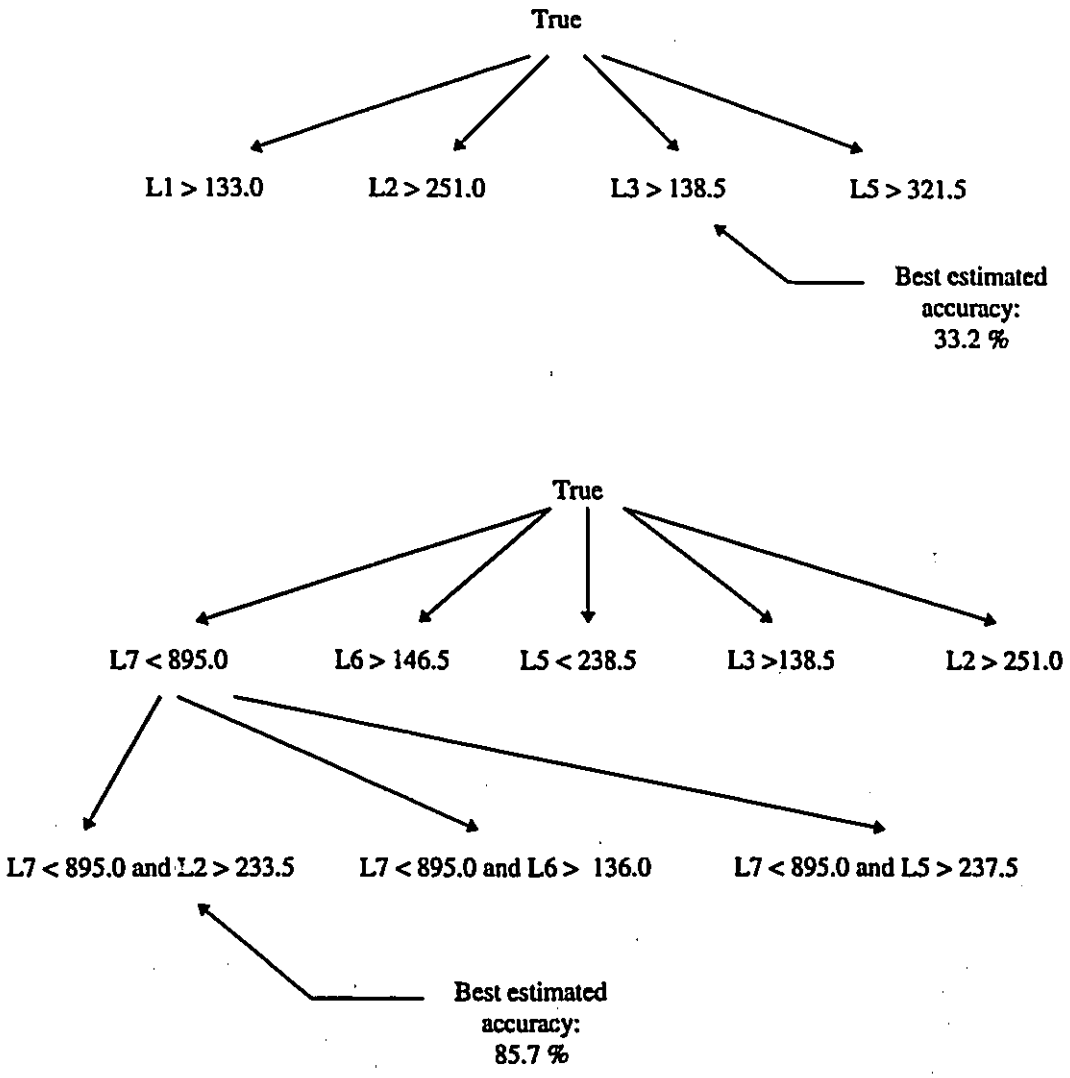


Figure 5.2 Example of a search expansion

The low expected accuracy of the explainable specialisations is due to the fact that all the examples of the training set they would normally cover, have already been eliminated by previous runs of the procedure FindBestCondition. The top level control procedure uses a greedy strategy to obtain a hopefully complete coverage of the set of positive examples and calls this procedure to find the rules which covers best the remaining positive examples. The positive examples which are easily covered by the explainable specialisations are thus removed after the induction of the first (explainable) rules. As a consequence, in the current situation (this one of the last runs of the procedure FindBestCondition), and because the model is incomplete, what remains are mainly positive examples which are not covered adequately by the available and explainable specialisations. These specialisations thus cover few positive examples and some negative examples, resulting in a poor estimated accuracy. Here, we see that the search expansion is motivated by the lack of explainable specialisations which cover adequately the remaining positive examples.

The second tree shows the states of the search after expansion at depth 1 and 2. At depth one, 8 non-explainable specialisations were added to the 4 explainable ones and the best five of the 12 were kept (only the specialisations that were kept are shown in the figure). At this depth, the best candidate rule is the rule corresponding to the specialisation L7 > 895.0, i.e.:

if L7 < 895.0 then L7 will increase (L7 is the target parameter)

This rule (shown above) has an expected accuracy of 54 %.

At depth 2, 42 specialisations of the 5 candidate rules in the beam are tested. Only 3 improve the example coverage of their direct ancestor and are kept. The best candidate rule at depth 2 is:

if L7 < 895.0 and L2 > 233.5 then L7 will increase

The expected accuracy of this rule is 85.7 % and is thus better than the best rules at depth 1 and 0 (at depth 0 the best rule is the default rule which predicts the most common class in the training data. In this case its predicted accuracy was 59.1 %). At depth 3, 10 specialisations are tested but none of them improves the coverage of their ancestor. The search thus stops, and the best candidate rule found is returned by the procedure. Here, the best rule is the best one found at depth 2.

The benefit of the search expansion appears clearly in this example. Without the expansion, the search would have stopped at depth one and the default rule would have been returned, resulting in no rule added to the ruleset to predict the class increase. Instead, a rule with a high expected accuracy was found by allowing non-explainable rules to be considered. Furthermore, the threshold algorithm biases the search for explainable rules at first, thus it ensures that they are favoured and that the qualitative model is fully exploited before allowing the search to expand.

As a summary, the main goal of this method is to detect when the bias for explainability has negative effects on the accuracy and, when it happens, to allow non-explainable rules to be induced if they perform better.

5.1.3 Effect of the threshold value

The variable named 'threshold' in the algorithm described in the previous section is a user-defined parameter. Its value reflects how confident the user feels in the qualitative model, as opposed to statistical evidence found in training data. The lower the threshold, the more we tend to trust the fully specified rules extracted from the model.

If threshold value is low, then we impose a low minimum expected accuracy on the explainable rules. As a consequence, search expansions will be rare, since the candidate explainable specialisations will most of the time be evaluated as good enough. Thus, we will have highly explainable rules. Explainability is favoured again accuracy.

If the threshold value is high (close to 1), search expansions will take place more often, resulting in potentially less explainable (but more accurate) rules. Accuracy is favoured against explainability (see Figure 5.3)

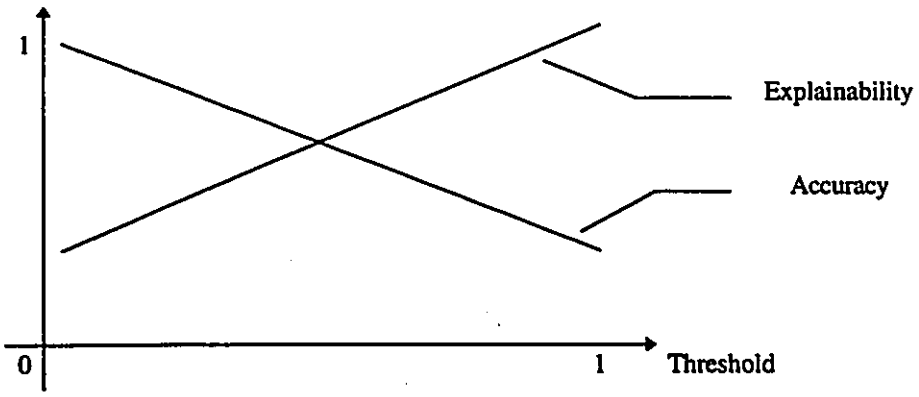


Figure 5.3 Qualitative variations of explainability and accuracy with the threshold value

In fact, the quantity (1 - threshold) can be seen as a measure of the confidence of the user in the qualitative model. Another way to view it is to consider the threshold value a bias of the learning system towards explainability, as opposed to accuracy.

5.1.4 Revisions of models using the threshold algorithm

When a model is severely incomplete, we expect search expansions to take place to induce rules corresponding to the missing knowledge. As a result, we will learn rules which cannot be explained by the qualitative model, since they do not correspond to any combination of pairs of fully specified rules.

Thus, if an important relationship is missing in the qualitative model then probably a significant part of the non-explainable rules induced after search expansions will be related with this missing link. For example, if a link connecting two parameters is essential to the model and is missing, we expect to induce non explainable rules, which would be explainable if the link was included in the qualitative model. These rules are evidence of a

correlation in the training data which cannot be explained in the qualitative model. Now, it is possible to use these rules to try to complete our model. In fact, these non-explainable rules can be seen as suggestions based on statistical evidence found in the data.

To suggest completion of the qualitative model, we proceed as follows:

- each time a search expansion occurs and a non-explainable specialisation of the candidate rules in the beam is selected as the new best rule so far, we store the corresponding specialised rule schema in a file. Note that non explainable rules generated after a search expansion are interesting because they have been obtained by pure induction from data. Thus, whenever one of these non explainable rules is the best rule in the beam after a search expansion, we expect that it contains information (induced from the data), explaining why the explainable rules were not performing well in the first place. For example, if the best rule in the beam after a search expansion is:

if p1 > 45 and p2 < 135 and p3 > -223 then target will increase

and if this rule is not explainable, we store the rule schema below:

if p1 > _ and p2 < _ and p3 > _ then target will increase

The interest of storing the rule schema rather than the rule itself comes form the fact that we are not interested by the numeric values but rather by the form of the rule itself. The rule schema is all we need to compare the rule against the qualitative model for possible revisions. Moreover, two rules which differ only in their numerical values contain the same information, from the point of view of explainability. In this case, storing the rule schemas will clearly indicate their similarity.

- once the induction is over, we process all the rule schemas stored. The goal of the processing is to normalize the rule schemas and to count the number of occurrence of each rule schema. As already pointed out for the fully specified rules extraction, the two rule schemas

if p1 > _ and p2 < _ then target will increase

if p1 < _ and p2 > _ then target will decrease

are considered as the same, in the sense that they are derived by the same piece of knowledge. If one of them is explainable then the other one is also explainable. The second one is just the translation of the first one for the class decrease: all the tests and the predicted class are inverted. Thus, the normalization consists only in inverting the rule schemas predicting a decrease of the target parameter so that they predict an increase of the target.

- we order the generated list of rule schemas from the one which occurred most often to the one which occurred the least often. Here is an example of list of rule schemas obtained through this process (to make it shorter, each rule schema was reduced to its tests. Thus "L6>" actually corresponds to the complete rule schema "if L6>_ then L7 (the target) will increase"):

Rule Schema	Frequency
L6>	0.203
L6> L7<	0.150
L7<	0.126
L5> L7<	0.087
L3> L5> L7<	0.055
L2> L3> L4<	0.049
L2> L7<	0.040
L3> L4< L5>	0.035
L3> L4<	0.030
L1> L3> L4<	0.023
L1> L5> L7<	0.014
L3> L4< L5<	0.013

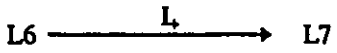
L4< L6>	0.010
L3> L7<	0.009
L3> L5< L6>	0.009
L2> L5> L7<	0.009

...

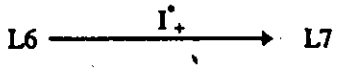
This list was obtained for the water pipes model and for a version of the qualitative model where the I*+ from L6 to L7 was removed. The threshold algorithm was run about two hundred times with different values of the threshold ranging from 0.1 to 0.9 to generate this list. 1930 non explainable rule schemas (including duplicates) were collected.

By examining the processed rule schemas, it is possible now to make the qualitative model more complete. If a rule schema appears often or if a particular parameter test appears in most of the obtained schemas, we can use the information to try to improve the model.

Returning to our example, the rule schema corresponding to "L6>" ("if L6>_ then L7 (the target) will increase") is induced most often. From this we can infer that L6 has an effect on the target which cannot be explained by the model. This is also confirmed by the second most often suggested rule schema "L6> L7<". Plausible completion of the qualitative model would then be the link



if we consider the first schema alone, or even better



which make the two first schemas be explainable.

A suggested rule schema does not actually correspond to only one link in the qualitative model. The propagation and evidence paths do not appear in a rule schema. The only information that a suggested rule schema gives us is something like: there exists a

parameter with an influence on the target parameter which cannot be explained. We can thus complete the model by adding a link from the parameter to the target and corresponding to this unexplained yet influence. This completion is a shorthand of what an expert would say, in the sense that it does not include any other parameter and that it does not try to use any already existing path to the target.

The main goal of the suggestions is to give advice on which parameters should be considered to improve a weak model, should the occasion arise. In other words, the suggestions point out the local weaknesses of the model, as evidenced by the training data. The suggestions will thus be very useful whenever a model will perform poorly (when the corresponding model-driven ruleset have a poor predictive accuracy) as a guide for the expert in charge of revising the model.

Another attractive possibility is to use the suggestions to build a model from scratch for a given physical system. We start with an empty qualitative model and we run the threshold algorithm, which in this case will be a purely data-driven induction. Then, the initial model will consist of a single link corresponding to the most often induced rule schema. We run the threshold algorithm again to obtain new completion for the initial model, then add a new link and so on. The iterations stop when no significant improvement is accomplished by adding a link, i.e. the rules induced with the totally constrained search have the same predictive accuracy than the rules induced with the standard CN2 algorithm.

This approach has some limitations. We will mainly obtain a star shaped qualitative model: every parameter will be connected directly to the target and very rarely to another parameter. The reason for that is the lack of information about how the state of a variable influences the target's behaviour. With the schema "if X > _ then Y will increase", we can do no more than insert an I+ link from X to Y. We do not actually know how the influence works (through which other variables and other links). While this type of model actually gives a direct view of each parameter's influence on the target, it does not contain any information about the internal connections and dependencies of the system. It could

not be directly used, for example, to try to understand in depth how the system works. Thus, we can say that 'an induced model' would probably contain less information than one designed by an expert with a perfect understanding of the physical system.

Finally, the suggestions can also be used to rectify incorrect models. If our model contains a incorrect relationship in the sense that it does not model accurately the actual behaviour of the physical system, then suggestions can be used to detect it easily by comparing the predicted influence of its nodes to the actual influences in the data, as revealed by the purely induced rule schemas.

We described the three possible ways to use suggestions gathered during the learning phase: completion, correction and building of a qualitative model (note that the building is actually a special kind of completion). We expect these approaches to be very useful when the need for a revision of the model occurs, because it actually points out the weaknesses of the model and guide the expert in its task. This is possible because the threshold algorithm can switch its bias during search.

The same results could not be obtained by using directly rules induced by CN2. As we have already noticed earlier, for the domain of the water tanks, for which we consider our model as being reasonably complete and correct, only 50% of the rules induced by CN2 are explainable. If our model was incomplete, we would have for example only 30% of explainable rules. Now, the problem is to distinguish between the 20% of rules which we consider useful and the 50% of rules which are mainly noise among the 70% which our incomplete model cannot explain.

The threshold algorithm has the advantage of biasing the search for explainable rules at first. Thus we induce fewer non-explainable rules. Furthermore, only non explainable rules are induced, which we expect to have a higher predictive power than the explainable rules. Hence, the suggestions will probably be more useful than the set of non explainable rules resulting from a learning with the standard CN2 algorithm.

5.2 Representing uncertainty

As we have already noticed in section 4.2.3, the expert might not be totally confident in the domain knowledge which he wants to include in the qualitative model. He might also be confronted with two or more different possible modelisations of a particular aspect of the physical system. Those are two situations when the expert has to choose between two imperfect solutions:

- if the expert includes the uncertain knowledge in the model, this knowledge will be considered as certain as the other parts of the model, when used for learning.

- if the uncertain knowledge is not added to the model, then rules explainable by this knowledge won't be considered during learning (with the totally constrained search).

We can bring a solution to this dilemma by allowing uncertainty at the design level. This way, the expert is able to express how confident he feels about relations in the qualitative model. We also want this uncertainty to be taken into account during the learning phase to bias the search to favour more certain parts of the knowledge against uncertain ones.

In this section we will describe our extension of the language used to build qualitative models, so that it allows a measure of certainty of each of the model's component. We will also show how this measure of uncertainty can be included in the search heuristic to bias the search in favour of the more certain concepts.

5.2.1 Uncertainty in the qualitative model

As described in chapter 3, the expert's knowledge is used to build a qualitative model of the physical system under study. The expert's role consists particularly in choosing qualitative relationships between the system's parameters to modelise as closely as he can its causal behavior.

We thus see that most of the knowledge expressed in the model is contained in the presence (or the absence) of the relationships themselves. It is actually when choosing whether a particular relationship is to be included in the model or not, that uncertainty may appear. It is thus natural to associate a measure of certainty of the knowledge included in the model with each link.

In order to allow a representation of uncertainty in our qualitative models, a belief weight is associated with each link included in the qualitative model. These weights take values between 0 and 1. A belief weight equal to 1 denotes an absolute certainty, while a belief weight equal to 0 denotes that the expert is certain that the link should not be included in the qualitative model. By definition, a link which is not in the model is considered to have a belief weight equal to 0. The belief weight represents how confident the expert is in the associated link. Based on the belief weights of the relationships in the qualitative model, we can now define the belief weight of a fully specified rule. We consider that any reasoning based on two links for example should have a belief measure equal to the minimum of the belief weights of the two links. The weakest link determines the weakness of the association of the two links. Thus we define the belief weight of a fully specified rule as the minimum of the belief weights of the corresponding relationships in the qualitative model. Note that it includes the links used for indirect evidence of a non-observable parameter or in the propagation path to the target. Also, the belief weight associated with a combination of two fully specified rules is the minimum of the two belief weights associated with each fully specified rule.

During the learning phase, induction starts with the most general rule (if true then target will increase/decrease). The most general rule is then specialised and we obtain candidate specialised rules, which are evaluated on the training examples. The best candidate rules are kept in the beam to be specialised further, and so on, until the hill-climbing search terminates. During specialisation, fully specified rules extracted from the qualitative model are used to generate the list of explainable specialisations of the given candidate rules in the beam. We associate with each explainable specialisation a belief weight equal to the

belief weight of the fully specified rule (or combination of fully specified rules) which explains the specialisation. It can happen that, for a given candidate rule, a specialisation can be explained by more than one fully specified rule (or combination of fully specified rules). In this case, the belief weight associated with the specialisation is the maximum of the belief weights of the fully specified rules (or combinations of fully specified rules) which explain the specialisation. This can be seen as having more than one explanations for a given fact, each of the explanation having a confidence value. Our strategy consist in retaining the best explanation and associating its confidence value to the fact.

The belief weights are also compiled in the lattice along with the corresponding specialisations. Thus, whenever a list of specialisations is extracted from the lattice during the inductive process, it also contains the associated belief weights.

5.2.2 Handling uncertainty during learning

The algorithm we use for learning here is a modification of CN2, which takes the belief weights of the explainable specialisations into account. When trying to find the best rule covering a set of positive examples, CN2 specialises a set of candidate rules, then keeps the best ones to be specialised further.

As opposed to the totally constrained search, we do not restrict here the specialisation operation to the list of specialisations contained in the lattice. All the specialisations are considered, but those which are not in the lattice have an associated belief weight equal to 0. By definition, the belief weight of a rule is the minimum of the belief weights of its attribute tests.

In CN2, the search heuristic is based on Laplace expected accuracy and gives a measure of a candidate rule's value. This value is used to sort the candidate rules and select the best ones to be specialised further. In order to handle uncertainty and to bias the search to rules with higher belief weights, we use the new search heuristic given below:

$$Value(R) = \alpha.Acc(R) + \beta.Blf(R)$$

where

R is a candidate rule

Acc(R) is the Laplace expected accuracy of **R**

Blf(R) is the belief weight associated with **R**

α and β are two user defined parameters between 0 and 1 such that $\alpha = 1 - \beta$

Our new search heuristic is a linear combination of the belief weight and of the expected accuracy of the candidate rule. The user-defined constants α and β represents the respective importance of the measure of certainty and of the expected accuracy in our evaluation of a rule.

If β is close to 0 (thus if α is close to 1), then the expected accuracy is favoured in the heuristic. Thus the accuracy is favoured during learning. Actually, if $\beta = 0$ ($\alpha = 1$), the behavior of the algorithm will be exactly the same as CN2.

If β is close to 1, then a rule's belief weight becomes the measure of the rule's value. Rules with higher belief weights are favored. As a consequence, explainability is favoured too (remember that a non-explainable rule has a belief weight equal to 0). If β is equal to 1, then the selection of the rules is done by only considering the belief weights. Note that for high values of β, if the expert includes only links with belief weights equal to 1 in the model, then the algorithm is similar to the totally constrained search presented in chapter 4.

Since accurate rules are desired, β should not be too big. Already a value of β such as 0.1 gives a significant advantage to explainable rules, since the differences between the expected accuracy of candidate rules is often small. Depending on their belief value, explainable rules will have a bonus between 0 and 0.1 (the product of the belief value and of 0.1) which will help them to be selected in comparison with non-explainable rules. Since the maximum difference of accuracy of rules in the beam (explainable or not), is

typically smaller than 0.3, this bonus represents already a considerable advantage given to explainable rules.

The way uncertainty is handled allows the expert to include two different models of some aspect of the system, and by choosing adequate belief weight, to favor the one he consider more plausible.

Also, non-explainable rules (and explainable rules with low belief weights) are considered during the search. If they have a high predictive power, they might overcome their handicap and be selected.

Note finally that belief weights are really an extension of our notion of explainability. Instead of being 0 and 1, one can now consider the explainability of a rule to be its belief weight. The algorithm can either behave like CN2 or like the totally constrained search algorithm depending on the values of α and β. Thus, we have actually provided a mechanism to trade-off between explainability (as measured by the belief weights) and accuracy during learning.

Chapter 6

Experimental results

This chapter presents an experimental evaluation of the different approaches described in chapters 4 and 5. Since we are mainly interested in the effect of the background knowledge on the behaviour of the inductive tool, CN2 will often be taken as a reference for comparisons. In this chapter, for the sake of brevity, **TCS** refers to the totally constrained search algorithm presented in chapter 4. **THR** and **WGHT** refer to the approaches developed in chapter 5, i.e. respectively to the threshold algorithm and to the modification of CN2 which takes belief weights in the qualitative model into account during learning.

6.1 Experimental design

6.1.1 Description of the data sets

A data set was generated for each of the three applications domains by using numeric simulators. It is important here to highlight that, as a result, these data sets are independent from the corresponding qualitative models. This is a fundamental difference with KARDIO [Bratko et al., 1984], where the qualitative model of the heart itself was used to generate the training examples for learning.

Examples in data sets are snapshots of the state of the physical system at a time T, i.e. the values of the observable parameters at time T, augmented by the class ('increase' or 'decrease') stating whether the target parameter was observed to have increased or decreased by tine T+ΔT. ΔT was taken equal to 1 second real-time for the water tanks

system, 1 minute real-time for the ore grinding system, and 0.5 second real-time for the analog electric circuit.

In order to generate random but still plausible states of the real physical system, simulations were conducted as follows:

- for the water tanks systems, the controllable parameters were randomly perturbed at intervals for approximately 50 seconds real-time (500 time-steps). Then the perturbations were stopped and the sates of observable parameters were recorded. The simulation continued for approximately 1 second to observe the evolution of the target parameter. These measures formed one example of the data set. The entire process was repeated 400 times to generate the whole data set.

- for the ore-grinding system, the same protocol was respected, except that the intervals were respectively 5 minutes and 1 minute real-time. The process was repeated about 540 times to generate the whole data set.

- for the analog electric circuit, the intervals were respectively 5 seconds and 1 second real-time and the process was repeated 200 times to generate the whole data set.

Table 6.1 Characteristics of the data sets

Application Domain	Number of examples	Numbers of observable parameters
Water tank system	400	7
Ore grinding system	538	10
Analog electric circuit	200	9

Table 6.1 summarizes the essentials aspects of each data set. Note that the number of examples for the electric circuit domain is much smaller than the number of examples for the two other domains. The reason for this is that the learning task associated with the electric circuit is expected to be less complex than for the two other domains, i.e. the description, with prediction rules, of the behaviour of the target parameter depending on the values of observable parameter, is expected to be shorter.

Note also that no artificial noise was introduced. As a matter of fact, CN2 was built to handle noisy data and avoid overfitting of the training examples, so we expect our approaches to possess the same ability to cope with noise in the data. However, since no experiments were done to specifically verify this aspect, it is unclear how well our modifications of CN2 behave in presence of noisy data.

6.1.2 Experimental protocol

For each run of the learning algorithms, the data set associated with the application domain was split into a training set and a testing set. Rules were induced using the training data set and (except for the standard CN2) the corresponding qualitative model. The testing set of examples was used to evaluate the predictive accuracy of induced rules.

There are several independent parameters which control the behaviour of the search and some of them are specific to a particular approach. CN2's parameters, the beam width and the depth limit, are shared by all the algorithms we present. See section 4.1 for a presentation of CN2 and for more information about the role of each parameter. The threshold algorithm (THR) has one specific parameter, namely the threshold value. Similarly, the second approach we presented in chapter 5 (WGHT) and which handles uncertainty, has one specific parameter which is β[1]. β represents the relative importance given in the search heuristic to the belief weight associated with a rule, as opposed to its predicted accuracy. The last independent parameter is also shared by all the approaches

[1] Since $\alpha = 1 - \beta$, α and β represent actually only one *independent* parameter.

and is the proportion of the data set which was used for training. The minimum and maximum proportions of examples from the data set used for training in our experiments, were respectively 5% and 70%. We believe that beyond 70%, too few examples are left for testing, thus making the evaluation of the accuracy of the induced ruleset less reliable.

Our experiments vary the number of training examples, the beam width and the depth limit[2], as well as the threshold value and the value of β. For each given set of parameters, the experiments were repeated at least 10 times with randomly chosen training sets.

For each experiment, we recorded the CPU time needed for training, the classificational accuracy of the induced ruleset measured on the testing set, and the explainability of induced rules. Note that the explainability of a ruleset, as defined in chapter 4, is the percentage of rule which are explainable, among the rules from the ruleset.

6.2 Results

6.2.1 Comparison of TCS with CN2

In this section, we present an experimental comparison of the totally constrained search, as described in chapter 4, with CN2. The aim of this comparison is to show the effect on learning of using qualitative knowledge. For each application domain, the corresponding qualitative model presented in section 3.1.2 is used by TCS to determine the set of rules which are explainable with respect to this model. Then, during learning, TCS heuristically searches for rules with high predictive accuracy, and which are also explainable. In contrast, the standard CN2 algorithm does not take the qualitative model into account and thus consider both explainable and non explainable rules during learning.

[2] The variation of the results with the beam width and the depth limit is essentially a characteristic of CN2 and the study of this variation is beyond the scope of this thesis. However, in most of our experiments and in order to obtain more representative results, we made these parameters vary and presented an averaged result.

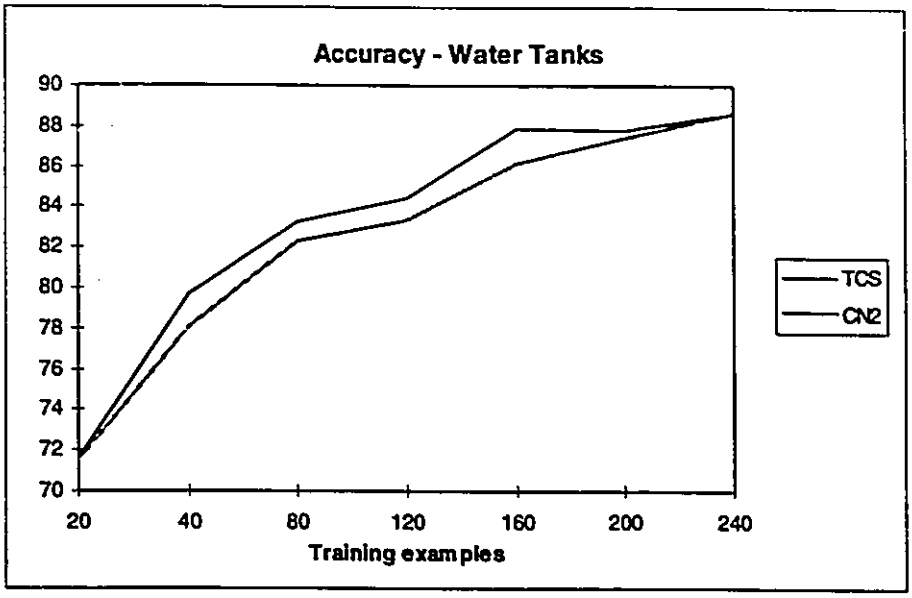


Figure 6.1

Figures 6.1, 6.2 and 6.3 show the predictive accuracy of rules learned by TCS and CN2 for the three different domains. For these experiments, the beam width was set to 5 and the depth limit was set to 3. For each training set size, the experiments were repeated 10

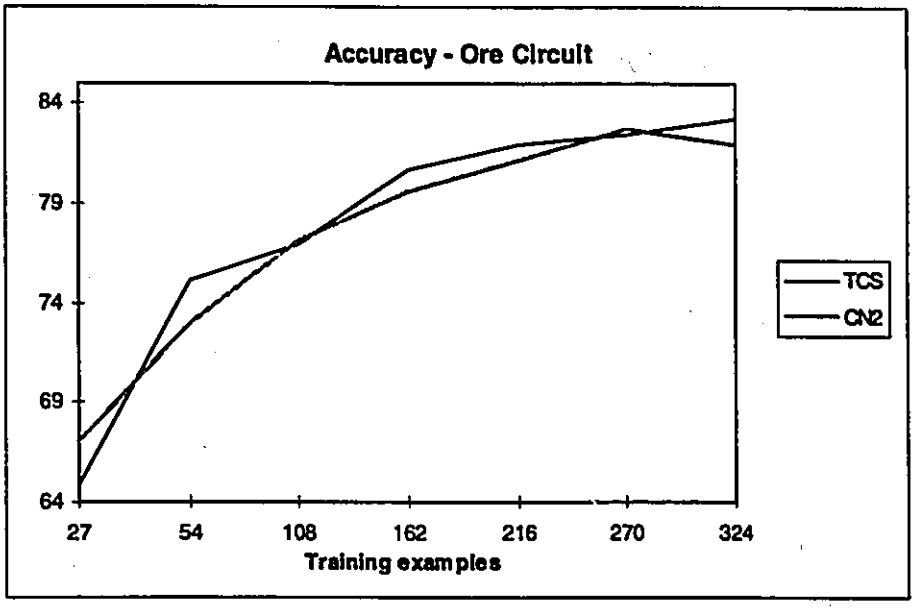


Figure 6.2

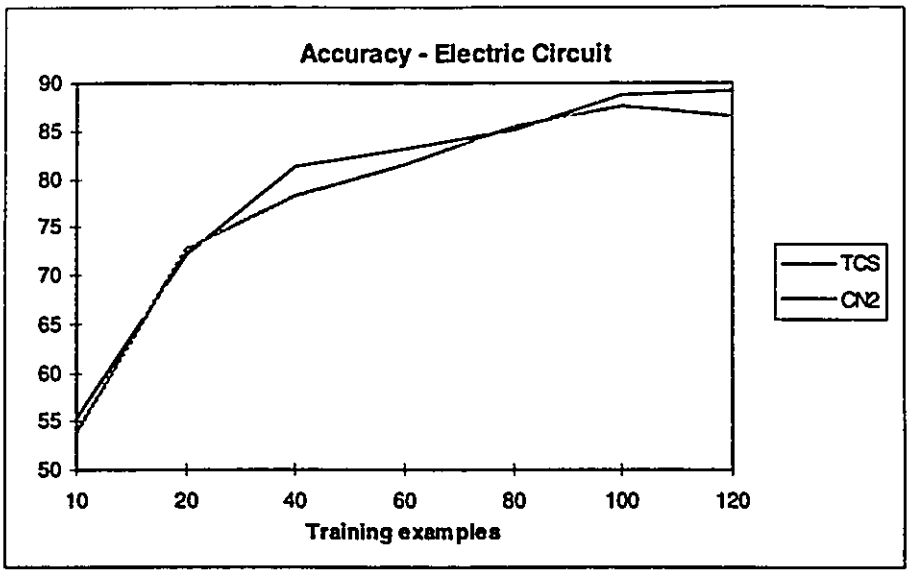


Figure 6.3

times and averaged. These figures are intended to give an example of the learning curves of TCS and CN2 and are less complete than the results of Table 6.2 presented below, since the experiments were done with fewer different values of the search parameters.

Table 6.2 contains the averaged results (and standard errors denoted by ±) of more intensive experiments. The two algorithms were run with depth limits of 2,3,4 (water tanks) and of 3,4,5 (ore circuit and electric circuit), and with beam widths 3,5 and 7 (water tanks) and of 5 (ore circuit and electric circuit). Three different training set sizes were used (30%, 50 % and 70% of the whole data sets). This represents a total of 540 runs for the water tanks domain and of 90 runs for each of the ore circuit and electric circuit.

These experiments show that the introduction of qualitative knowledge improves classificational accuracy of the learned rules in two of the application domains (ore-grinding system and electric circuit). For the water tanks system, the accuracy of rules obtained with TCS is comparable on average with the accuracy of rules induced by CN2. For the two other domains, considerable accuracy improvements are obtained, even

though the use of the qualitative model in TCS restricts the space of rules available during the learning phase. It thus tends to show that using a qualitative model helps to prune out rules, which by chance perform well on the training data set, but which have a poor classificational accuracy on unseen data. In this case, it is equivalent to say that the qualitative model injects useful knowledge into the learned rules and thus improves performance.

Table 6.2 Experimental evaluation of TCS and CN2

	Accuracy		CPU time (sec)		Explainability	
	CN2	TCS	CN2	TCS	CN2	TCS
Water tanks	86.69 ± 0.13	86.56 ± 0.14	17.53 ± 0.27	11.61 ± 0.41	49.5 % ± 0.6	100 % ± 0.0
Ore circuit	80.39 ± 0.31	81.53 ± 0.29	68.83 ± 3.72	50.16 ± 2.64	40.4 % ± 0.9	100 % ± 0.0
Electric circuit	85.61 ± 0.45	86.39 ± 0.46	11.42 ± 0.58	4.15 ± 0.22	50.2 % ± 2.0	100 % ± 0.0

Moreover, for the three application domains, the learning time is substantially reduced with TCS, as opposed to CN2. This is not surprising, since the qualitative knowledge is used to restrict the search space in TCS. During learning, the search has fewer possibilities to explore (less specialisations of the rules currently in the search beam to evaluate), resulting in less time needed to induce the ruleset. Note that compiling the qualitative model to extract the fully specified rules and build the lattice of explainable specialisations, involves an overhead which is not taken into account in the results of Table 6.2. However, since the model has to be compiled only once and since the time needed for this operation is very small compared to the average learning times, it can reasonably be neglected.

Finally, the most important benefit of TCS, as opposed to CN2 is the total explainability of the rules produced. As shown in Table 6.2, about only 40% to 50 % (depending on the application domain) of the rules induced by the standard CN2 algorithm can be explained with respect to our qualitative model of the physical system. In contrast, all the rules induced by TCS 'make sense' and can be easily explained with the qualitative model. In fact, we exploit this advantage by deriving causal qualitative explanation for each one of the learned rules (see section 4.2.2 for an example). The interpretability of the results, which we consider an essential aspect of learning, is thus considerably improved by the use of a qualitative model.

6.2.2 Evaluation of the threshold algorithm

The first approach presented in chapter 5, namely the threshold algorithm (THR), is intended to deal with incomplete or incorrect qualitative models. In the case where the model is very imperfect, we expect the rules produced by TCS to have a low classificational accuracy, as opposed to the rules produced by the standard CN2 algorithm. On the other hand, as described in the previous section, CN2 learns ruleset with a low explainability. The main goal of THR is then to learn rules with a good accuracy and with a higher explainability than the explainability of rules produced by CN2. Moreover, THR has an important parameter (the threshold value) which represents how strongly the search is biased to favor accuracy against explainability.

The first experiments presented in this section were conducted with the exact qualitative models presented in chapter 3 and are the same as those which were used to compare TCS to CN2. The intent here is to test the performance of THR in presence of good qualitative models. It is important to make sure that THR's performance is not significantly worse than the performance of TCS or CN2 in the 'good cases'.

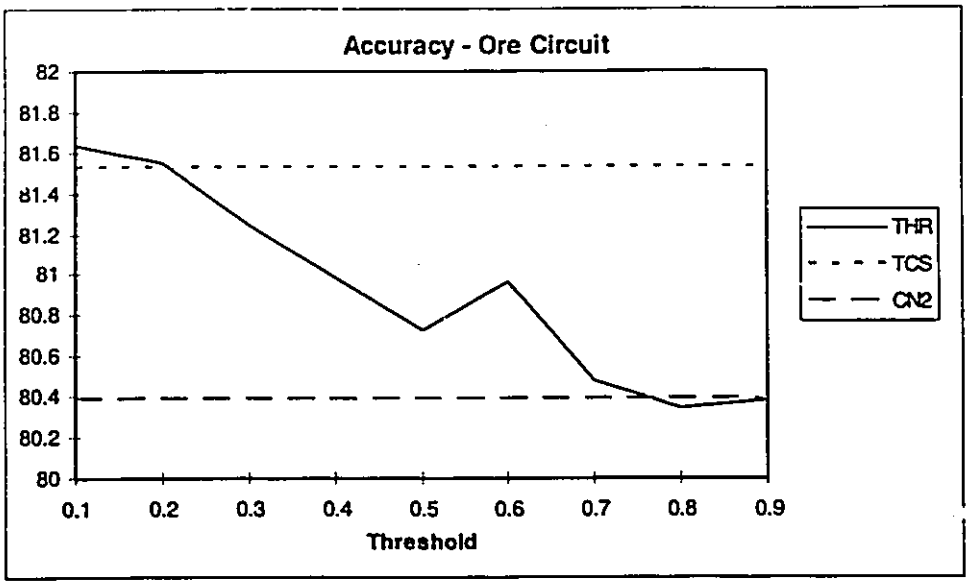


Figure 6.4

Figures 6.4, 6.5, and 6.6 show the accuracy, as a function of the threshold value, of the rules obtained by THR for the three application domains. To allow comparisons, the accuracy of the rules produced by TCS and CN2 in the same conditions are also indicated. Since the latter approaches are not affected by the threshold value, the resulting accuracies do not vary with this parameter and are plotted as horizontal lines.

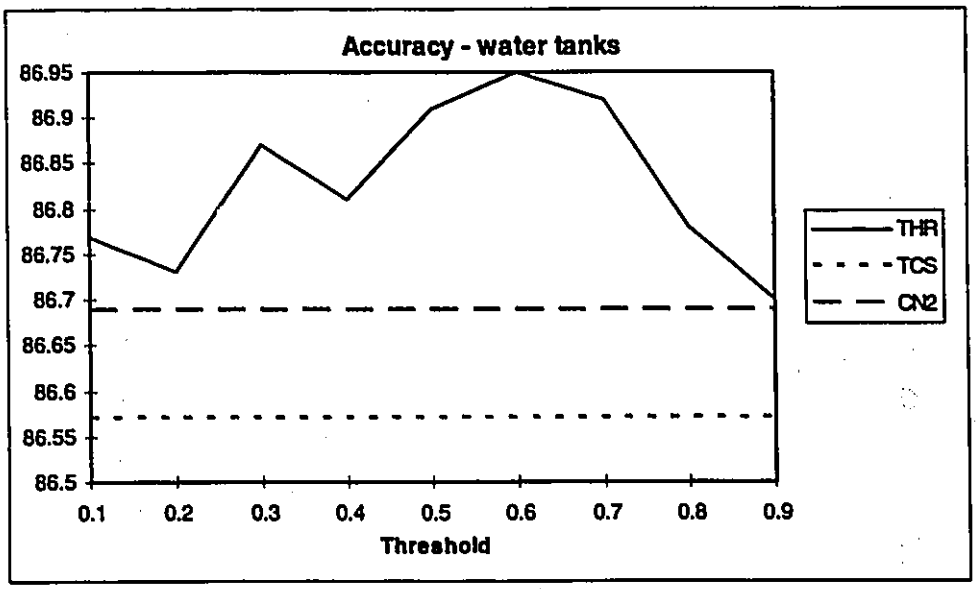


Figure 6.5

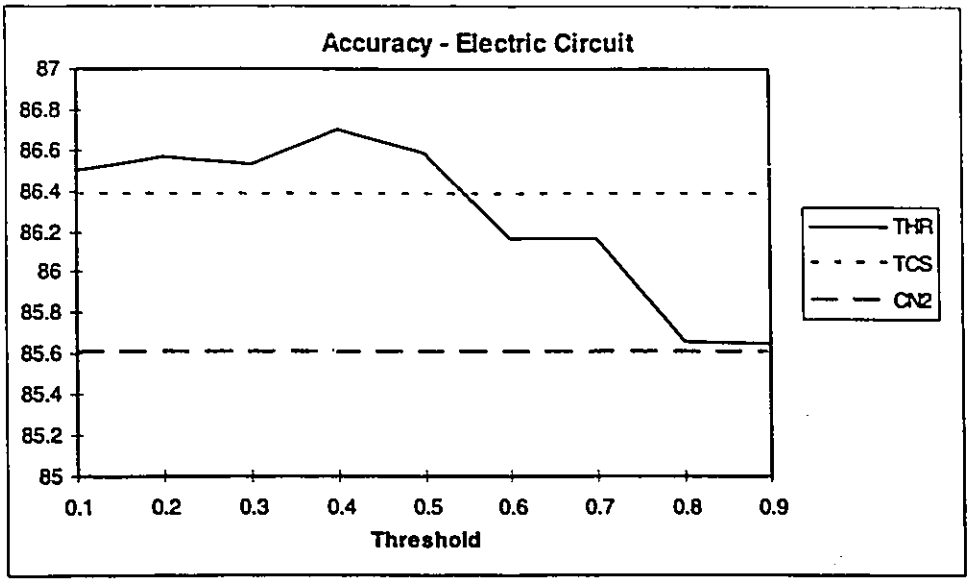


Figure 6.6

The experiments were conducted with the same parameter values than the ones of the previous section and whose results were presented in Table 6.2. In addition, for THR, the threshold value varied from 0.1 to 0.9.

The way THR was designed mixed the search strategies of both TCS and CN2. As special cases, THR can behave similarly to TCS (for low threshold values) or to CN2 (for high

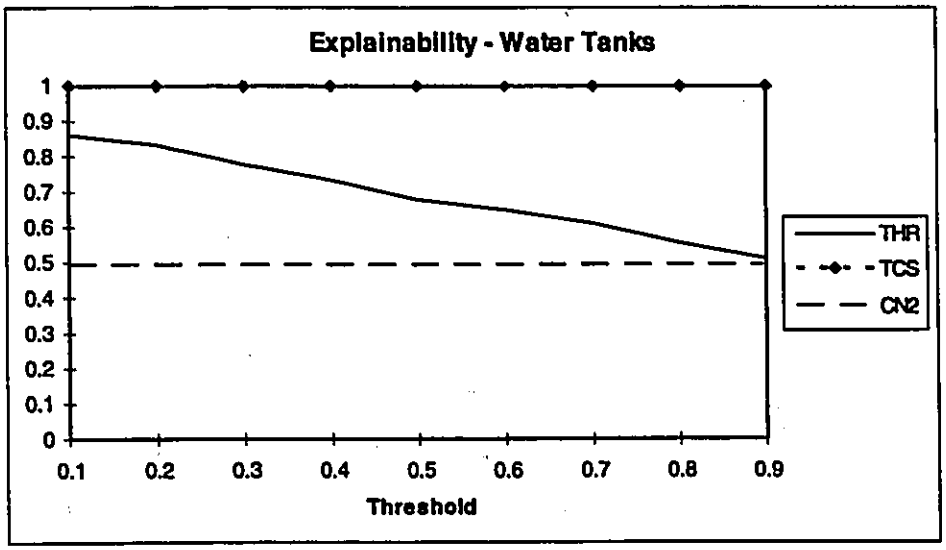


Figure 6.7

threshold values). It is thus legitimate to expect that the accuracy of the rules induced by THR will be included in the interval bounded by the accuracy of rules produced respectively by CN2 and TCS. This is indeed the case in our experiments for two applications domains, the ore-grinding system and the electric circuit. Surprisingly, for the water tanks system and for all the tested threshold values, the classificational accuracy of the rules learned by THR is higher than both corresponding accuracy obtained with TCS and CN2.

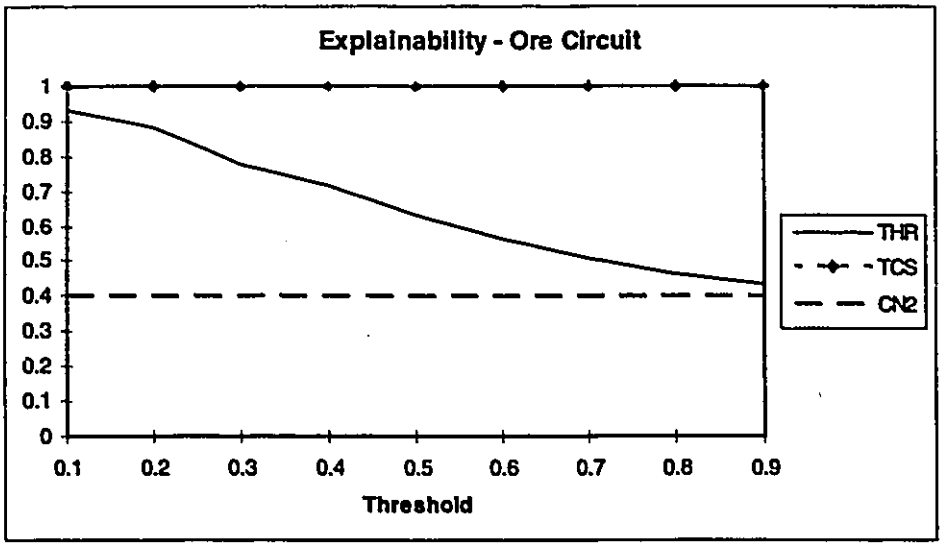


Figure 6.8

It thus seems to indicate that the combinations of both methods sometimes allow the hill-climbing search to overcome local maximums and to produce rules with a higher classificational accuracy. A plausible explanation of this phenomenon could be a slight weakness of our proposed qualitative model of the water tanks system. This hypothesis is strengthened by the fact that TCS obtains a slightly lower accuracy than CN2 for this application domain, whereas TCS performs better in the two other domains. A slight increase of the accuracy was also observed with low values of the threshold for the electric circuit domain.

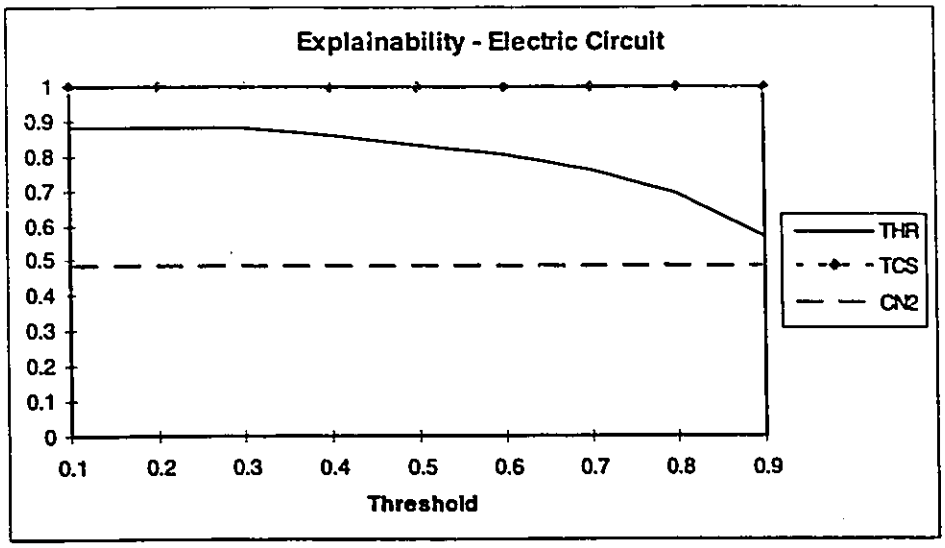


Figure 6.9

The explainability of the rulesets produced by THR in the same experiments were also recorded and are shown in Figures 6.7, 6.8 and 6.9 for the three applications domains. In each case, the explainability of the rules learned by THR decreases monotically with the threshold value, from a value close to 100 % down to the explainability of the rules induced by CN2 for the domain. This behaviour was expected since with high values of the threshold, search expansions occur more often, thus augmenting the possibilities for the learner to consider non explainable rules during search.

Note that even with a threshold value equal to 0.0, THR does not necessarily learn rulesets which are totally explainable. Since we allowed the search to expand when the set of explainable specialisations of the rules currently in the search-beam is empty, THR may learn non-explainable rules for any value of the threshold.

As a conclusion for this first set of experiments, we thus see that THR has an acceptable behavior in presence of reasonably good qualitative models. The accuracy of the produced ruleset is pretty much comparable to accuracy obtained with TCS and CN2 (and sometimes higher than both). Furthermore, as expected, the explainability varies

monotically with the threshold between 100% and the explainability obtained with CN2 for the corresponding domain.

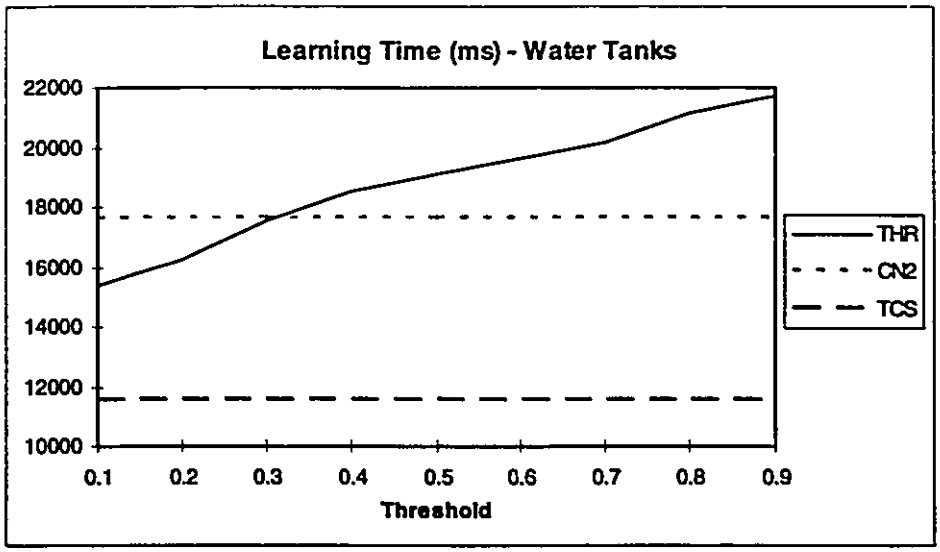


Figure 6.10

The learning times corresponding to these experiments are shown in Figures 6.10, 6.11, and 6.12. In both the water tanks and ore-grinding circuit domains, low values of the threshold make THR learn in a time similar to TCS. The learning time then increases monotically with the threshold value and becomes higher than the learning time of CN2 for the threshold values above 0.4. For the electric circuit domain, the learning time also increases with the threshold value, but remains smaller than CN2's learning time in the same conditions.

The rise, with the threshold value, of the time spent by THR to learn, can be easily explained. Our implementation of THR involves some backtracking each time a search expansion occurs. Thus, the amou.. of time spent to backtrack increases with the threshold value and causes THR's learning time to exceed CN2's learning time.

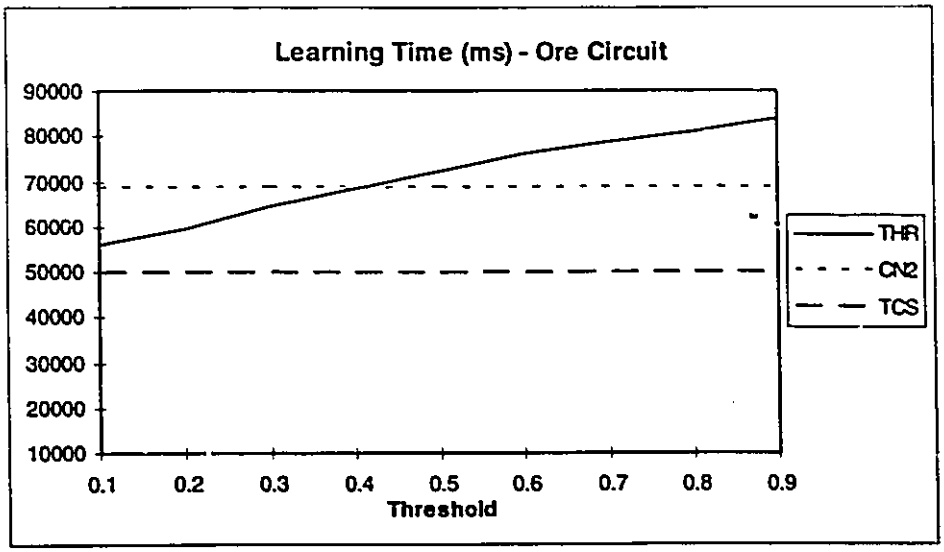


Figure 6.11

In order to evaluate the behavior of THR in presence of an incomplete qualitative model, three different experiments were conducted. The first one, we will call it **WT1,** consisted in removing the I*+ link connecting L6 to L7 in the qualitative model of the water tanks system (see Figure 3.3). Note that, as a matter of fact, this was equivalent to the removal of the three links along the path L1, L4, L6 and L7, since L6 and L4 are actually cut from the target. For the second one, **WT2,** *all* the links from the qualitative model of the water

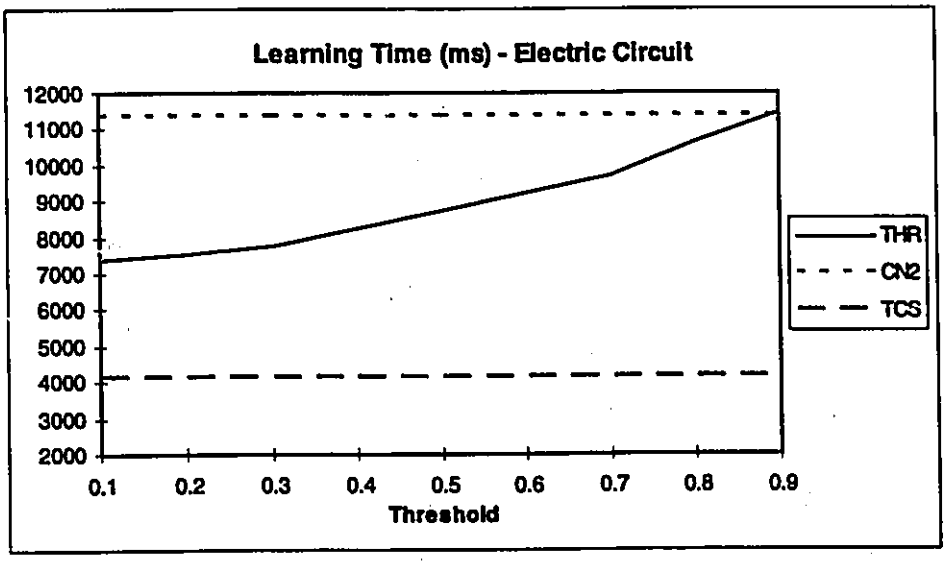


Figure 6.12

pipes were removed, except for the I+ link connecting L5 to L7. The last experiment, **EC1**, consisted in removing the I*- link connecting V8 to V9 (see Figure 3.9).

For each of these experiments, THR, TCS and CN2 were run with the depth limit set to 3, the beam width set to 5 and 65 % of the data set was used for training. The experiments were repeated 20 times with different training sets (of the same size). In addition, the threshold value varied from 0.1 to 0.9. The results are shown in Table 6.3 and 6.4. Note that the first column of table 6.4 indicates the value of the threshold and also that the indicated explainability was measured with respect to the incomplete qualitative model.

Table 6.3 Effect of the incompleteness of the model on CN2 and TCS

	WT1		WT2		EC1	
	Accuracy	Explainabi.	Accuracy	Explainabi.	Accuracy	Explainabi.
TCS	81.1% ± 0.6	100% ± 0.0	70.7% ± 0.8	100% ± 0.0	82.7% ± 1.0	100% ± 0.0
CN2	88.0% ± 0.5	9.4% ± 0.6	88.0% ± 0.5	0.0% ± 0.0	88.6% ± 0.9	25.0% ± 2.8

Table 6.4 Effect of the incompleteness of the qualitative model on THR with various threshold values

	WT1		WT2		EC1	
	Accuracy	Explainabi.	Accuracy	Explainabi.	Accuracy	Explainabi.
0.1	84.5% ± 0.5	64.7% ± 1.1	87.7% ± 0.5	5.2% ± 0.4	86.5% ± 1.1	72.0% ± 2.4
0.2	85.4% ± 0.6	59.7% ± 1.6	88.3% ± 0.3	4.9% ± 0.4	87.0% ± 1.3	70.0% ± 2.2
0.3	85.9% ± 0.5	55.8% ± 1.1	88.2% ± 0.5	5.0% ± 0.4	87.4% ± 0.9	64.8% ± 2.1

0.4	86.7% ± 0.5	50.3% ± 1.1	88.5% ± 0.6	5.1% ± 0.4	88.1% ± 0.9	59.1% ± 2.2
0.5	88.1% ± 0.6	43.6% ± 1.0	88.5% ± 0.6	5.1% ± 0.4	87.7% ± 0.9	53.5% ± 2.0
0.6	87.8% ± 0.6	35.0% ± 1.0	87.9% ± 0.6	5.2% ± 0.4	87.4% ± 0.8	46.9% ± 1.9
0.7	88.3% ± 0.6	28.1% ± 1.4	88.3% ± 0.6	5.4% ± 0.4	88.4% ± 1.0	41.5% ± 1.6
0.8	87.2% ± 0.5	18.4% ± 1.4	88.2% ± 0.5	5.5% ± 0.4	89.3% ± 0.8	35.8% ± 1.7
0.9	87.8% ± 0.5	13.1% ± 0.7	87.9% ± 0.5	5.0% ± 0.4	88.4% ± 1.0	28.4% ± 1.5

The results shown in Table 6.3 illustrates well the effects of the incompleteness of the qualitative model on the learning performance of TCS. For WT1, TCS loses approximately 7% of predictive accuracy comparatively with CN2. It thus appears that, by removing links in the qualitative model, we also removed from the set of explainable rules, rules that had a high predictive accuracy and that were needed to classify examples. The explainability of the rules induced by CN2 illustrates this weakness of the model. The truncated model explains only 9.4 % of the rules produced by CN2, whereas the original model was able to explain about 50 % of this rules. Also, and this is not shown in Table 6.3, during learning itself, the classificational accuracy on the *training* data of the ruleset produced by TCS was only of 85.9 %, while it is usually around 95 % for this application domain with the original model.

WT1 is thus a good example of a situation where TCS's bias for total explainability leads to poor performance, and this because of a strong incompleteness of the qualitative model. Results collected from the experiment EC1 are very comparable: substantial loss of accuracy for TCS and diminution of the explainability of CN2's results. In the case of WT2, the same effects were also observed but with a greater amplitude since the incompleteness of the model was more important. TCS loses almost 20 % of accuracy and

the explainability of the rules induced by CN2 shows that CN2 never selects rules which are explainable with respect to the truncated qualitative model in its search heuristically guided for accuracy.

While they reassuringly show that the qualitative knowledge provided to TCS has indeed a strong impact on its learning performance, these experiments are typical of situations where neither TCS nor CN2 gives satisfying results. TCS's classificational accuracy is greatly affected by the incompleteness of the model and, on the other hand, CN2 gives poorly explainable results. Note that CN2's results are independent of the model and the loss of explainability is due to the incompleteness of the model. Nonetheless, as shown in the comparison of TCS with CN2, since the latter is not biased at all for explainability, it does not naturally produce rulesets with a high explainability. It is thus legitimate to expect, that a search strategy with a weak bias for explainability would be able to produce rulesets with an accuracy comparable to the accuracy of rules induced by CN2 and also with a better explainability. The study of the results obtained with THR on the same incomplete models confirms the hypothesis that it is possible to exploit more efficiently the qualitative knowledge provided to obtain a better explainability than with CN2, while maintaining the accuracy.

Table 6.4 shows the experimental results obtained, in the same conditions, by using THR. The threshold value was varied between 0.1 and 0.9. For WT1, the accuracy obtained tends to increase with the threshold value. The minimum, 84.5% for the threshold set to 0.1, is significantly higher than the accuracy obtained with TCS. Also, for threshold values greater than 0.4, the obtained accuracy is comparable with the accuracy obtained with CN2. On the other hand, the explainability (with respect to the truncated model) decreases monotonically with the threshold value. The minimum, obtained with the threshold set to 0.9, remains higher than the explainability of rulesets induced by CN2.

While the experiment EC1 shows the same qualitative variations of the explainability and accuracy of the rulesets produced by THR, the results obtained with WT2 are noticeably

different: the accuracy and explainability of the induced rules do not significantly vary with the threshold value. The accuracy is about the same as the one obtained with CN2, and the explainability is higher, around 5 %.

From these experimental results, we can derive two conclusions about THR's behaviour in presence of incomplete qualitative models. First, when incompleteness of the model is important but still reasonable (WT1 and EC1), THR trades off between TCS and CN2's behaviors. For low values of the threshold, THR produces results closer to TCS and for high values of the threshold it produces results similar to CN2. Thus, if the threshold increases, non explainable rules are induced more often, but with the benefit of increasing the overall classificational accuracy of the produced ruleset. In this case, we see that it is thus possible to obtain a trade-off between accuracy and explainability, as expected in chapter 5.

Second, when the model is dramatically incomplete (WT2), THR produces an accuracy similar to CN2 and slightly higher explainability. In this case, the qualitative knowledge is too poor to allow a real trade-off: the explainable rules are too sparse to allow a satisfying classification of the training examples and search expansions take place most of the time, resulting in an behavior very similar to CN2.

Surprisingly, for the three experiments, the learning times obtained with THR were usually smaller than CN2's learning time (except for values of the threshold like 0.8 or 0.9 for which they were comparable). It seems, in these cases, that the time saved by the use of an even incomplete qualitative knowledge, compensates the computational cost of the backtracking involved in search expansions.

As a conclusion of the tests with incomplete models, it thus appears advantageous to use THR, rather than TCS or CN2, because, for models which are not excessively incomplete, it allows a trade-off between these two methods. By trade-off, we mean that we can voluntarily decide to sacrifice explainability to gain accuracy or vice versa. We achieve this by tuning the value of the threshold in THR. In fact, our experiments showed that THR

can be used to obtain rules with the same accuracy and a much higher explainability than CN2. Thus, we do not actually obtain a trade-off but a real improvement of the results in comparison with CN2. Furthermore, THR does not appear to be more computationally expensive than CN2.

While this is not our principal goal, we have already pointed out in chapter 5 that THR can be used to help revisions of the qualitative model based on statistical evidence found in the training data. Each time a search expansion takes place during learning and a non-explainable specialisation of a rule in the current search-beam is the best rule found so far during the search, we store the corresponding normalized rule schema. These rule schemas illustrate an influence of one observable parameter on the target parameter and can thus be used as suggestions for plausible completion of the qualitative model, if the need arises. For the three experiments which were conducted with incomplete models, we collected the non-explainable rule schemas generated by the search expansions.

For WT1, the obtained list was already presented in chapter 3. Since the two most often induced non-explainable rule schemas were "L6>" and "L6> L7<", a plausible completion of the qualitative model consists in adding an I*+ connecting L6 to L7. In this case, we thus see, that the completion suggested by evidence found in the data by THR, corresponds exactly to the link that was removed from the original qualitative model.

For WT2, the top of the sorted list of most often induced non-explainable rule schemas is:

Schema	Frequency
L5> L7<	0.380
L3>	0.236
L6> L7<	0.070
L3> L5>	0.065

The first rule schema of the list suggests a transformation of the I+ link connecting L5 to L7 to an I*+ link. This link does not exist in the original qualitative model. The second rule schema of the list (L3>) suggest the addition of an I+ link connecting L3 to L7. This link does not exist in the original model but actually correspond to the influence of L3, propagated to L7 along the path L3, L5, L7. The third one suggests the addition of an I*+ link from L6 to L7 and corresponds exactly to an existing link in the original model. The fourth one (L3> L5>) does not suggest any completion since it can be seen as the combination of the influences of L5 to L7 through the I+ (or I*+, if modified) link and of L3 to L7 through the added I+ link. For this experiment, we thus see, that, even if not perfect, the information collected, by pure inductive means, during learning by THR can be of great help in the completion of an even severely incomplete qualitative model.

For EC1, the top of the sorted list of induced rule schemas is:

Schema	Frequency
V8>	0.395
V10< V4> V8>	0.147
V10< V11< V8>	0.069
V10< V8>	0.059
V4> V8>	0.047
V6<	0.039

Taken alone, the first rule schema suggests the addition of an I+ link connecting V8 to V11. This link does not exist in the original qualitative model but corresponds to the path V8, V9, V10, V11. This first completion is thus equivalent to the link which was removed from the original model. It also explains the first six rule schemas of the list.

In this last example, we can see that one of the limitations of the method of completion, as already discussed in chapter 5, is that it tends to create a star-shaped model: all the parameters are directly connected to the target. For EC1, putting back into the original model the link that was removed for the experiment also explains the non-explainable rule

schemas. However, in order to consider this completion, one would have to take into account the structure of the physical system itself. Given a blueprint of the electric circuit and the non-explainable rule schemas, an expert would immediately find the right completion. Without any other information but the qualitative model, the most natural way to complete the model is to link directly the parameter, whose influence is not explained by the incomplete model, to the target.

6.2.3 Evaluation of the representation of uncertainty

In chapter 5, we presented a method which is intended to handle uncertainty when designing the qualitative model. This method, WGHT, allows the expert to associate a belief weight with each link of the model, representing how confident he feels in this link. These weights are then used during learning to compute a belief value associated with each rule considered by the inductive search.

Furthermore, the search heuristic uses this belief value, as well as the expected accuracy measured on the training data, to determine the value of each rule. In other words, the search is biased to favor rules with higher belief values. The relative importance given to the belief value and the expected accuracy in the search heuristic is represented by the value of the parameter β.

We expect the benefits of this approach to be a greater flexibility at the design level of a qualitative model. For example, the expert is able to include two different modelisations of some specific parts of the model and to favor one against the other during learning. The second advantage is that non-explainable rules are considered during search. If non-explainable rules perform considerably better on the training data than the explainable ones, they may be selected. As a consequence, we expect WGHT to learn rulesets with high accuracy, and to also exploit efficiently the qualitative knowledge provided to generate highly explainable rules. Thus, WGHT allows a trade-off between accuracy and explainability of the results and constitutes in this sense an alternative to THR.

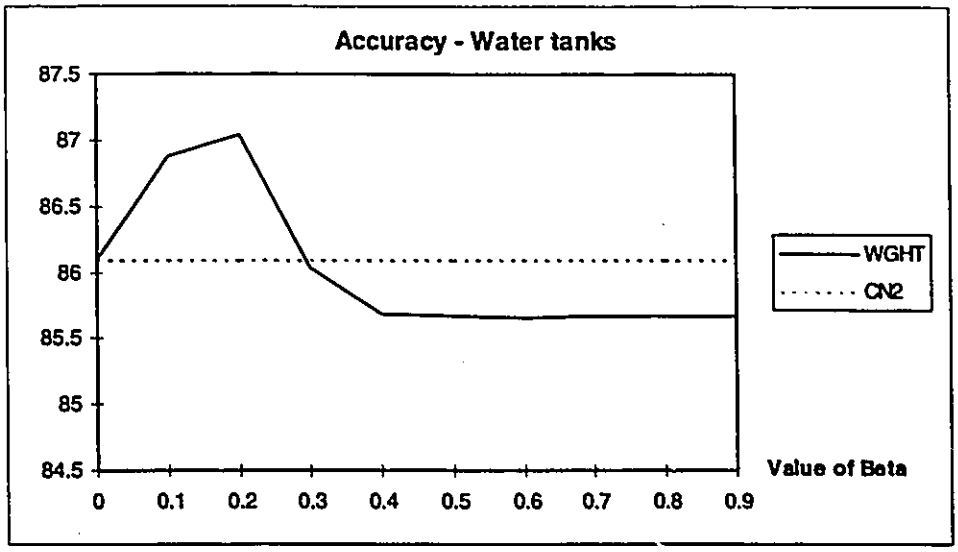


Figure 6.13

As for THR, we will first consider the experimental results obtained when 'good' qualitative models are provided to WGHT. In the next experiments, and for each application domain, we used the qualitative models described in chapter 3. A belief weight equal to 1.0 was associated with each link of the models. The qualitative knowledge provided to WGHT in these experiments is thus exactly the same as the knowledge which was provided to TCS or THR in the experiments of the previous sections: all the explainable rules have a belief value equal to 1.0, and thus none is preferred from the point of view of the belief value. Note however that WGHT differs from TCS because it will also consider non-explainable rules during the inductive search, but these rules will have an associated belief value of 0.0.

The experiments were conducted for WGHT and CN2 with the depth limit set to 2,3,4 (water tanks domain) and 3,4,5 (ore-grinding system and electric circuit), the beam width set to 3,5,7 (water tanks system) and 5 (ore-grinding system and electric circuit), and the training set size set to 30 %, 50 % and 70 % of the whole data sets.

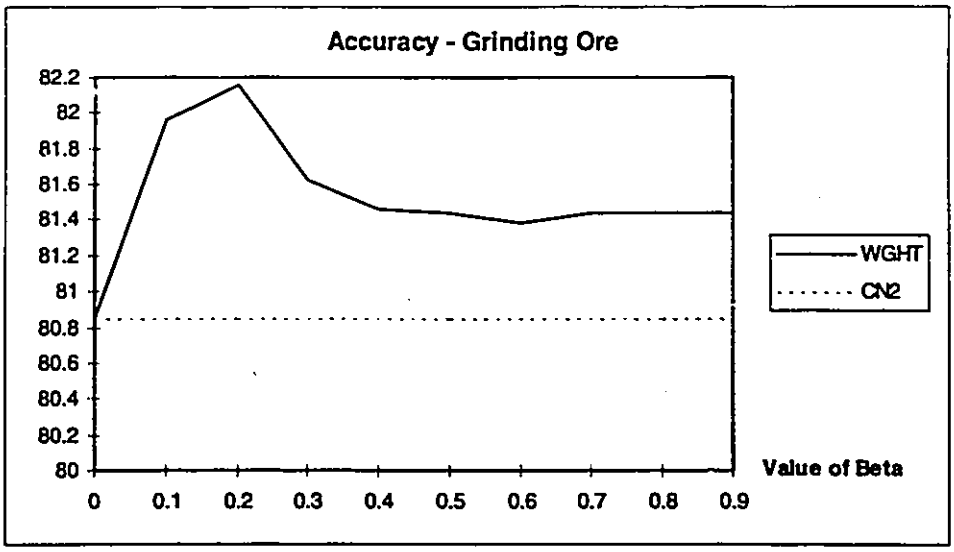


Figure 6.14

Figure 6.13 to Figure 6.15 show the accuracy of the rulesets learned by WGHT with various values of β ranging from 0.0 to 0.9. For comparison purposes, the accuracy obtained with CN2 in the same conditions is also indicated. Note that for β=0.0, WGHT behaves exactly like CN2 (the belief weights are ignored) and that for high values of β (β>0.5), the results obtained are similar to the ones obtained by TCS. When β becomes greater than 0.5, the non-explainable rules have such a disadvantage, when compared to the explainable rules, in the heuristic evaluation, that they are never selected. As a consequence, the search behaves as if it was restricted to the subspace of explainable rules, and produces exactly the same results as TCS.

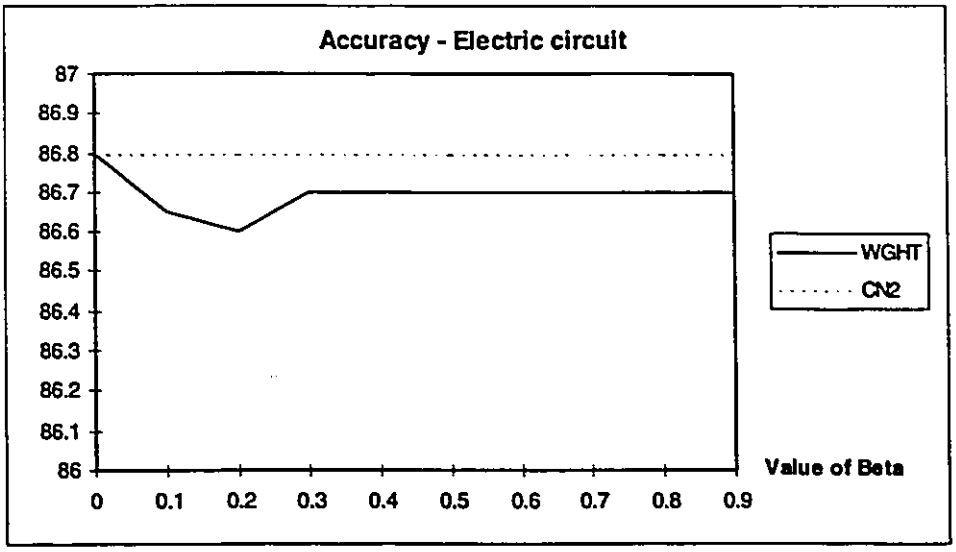


Figure 6.15

For two of the domains (the water tanks system and the ore-grinding circuit), the obtained accuracy is maximum for the value of β set to 0.2 and significantly better than both accuracy of CN2 and TCS (explicit reference is not made to the results obtained by TCS in the figures, but is equal in each experiment to the results obtained by WGHT with the value of β set to 0.9). This tends to suggest that, at least for these two domains, allowing

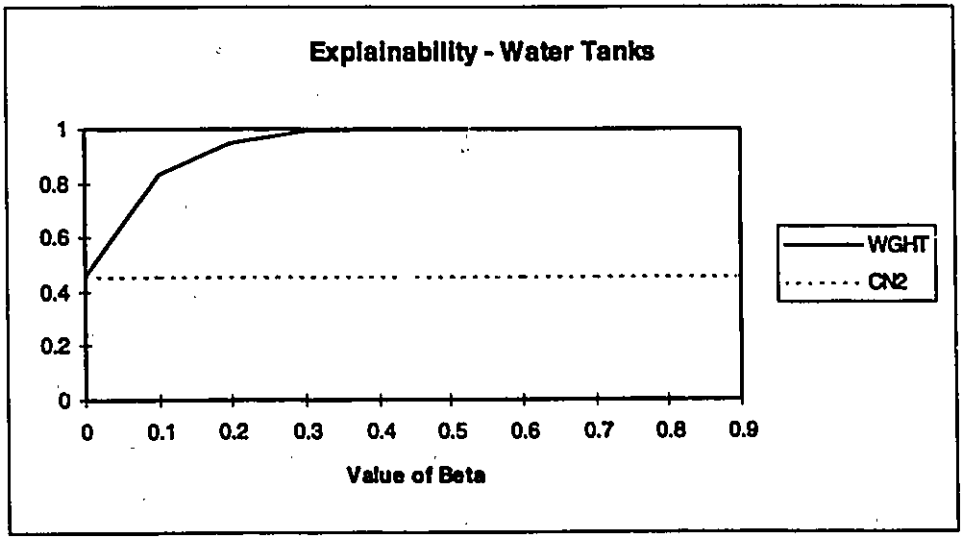


Figure 6.16

non-explainable rules to be selected when they have an expected accuracy at least 25 % higher than explainable rules, improves the learning performance.

Note also that, as expected, values of β greater than 0.5 makes WGHT behave identically to TCS. This observation holds for both the accuracy and explainability of the results. The learning time is higher with WGHT than with TCS in any case, since the former evaluates also non-explainable rules during the inductive search.

For the electric circuit domain, accuracy is not improved by using WGHT. A slight decrease, but not significant, was even observed. It appears that, when the performance of TCS and CN2 are very close, there is not much benefit in trying to combine both explainable and non-explainable rules. Nonetheless, experimental results show that, for the three application domains, the accuracy obtained with WGHT is similar to the accuracy obtained with TCS or CN2, and that it may be improved with low values of β.

Figure 6.16 and Figure 6.17 show respectively the explainability and the learning time of the rulesets produced by WGHT for the water tanks domain. The results obtained for the two other domains are not shown but were similar. As expected, the explainability

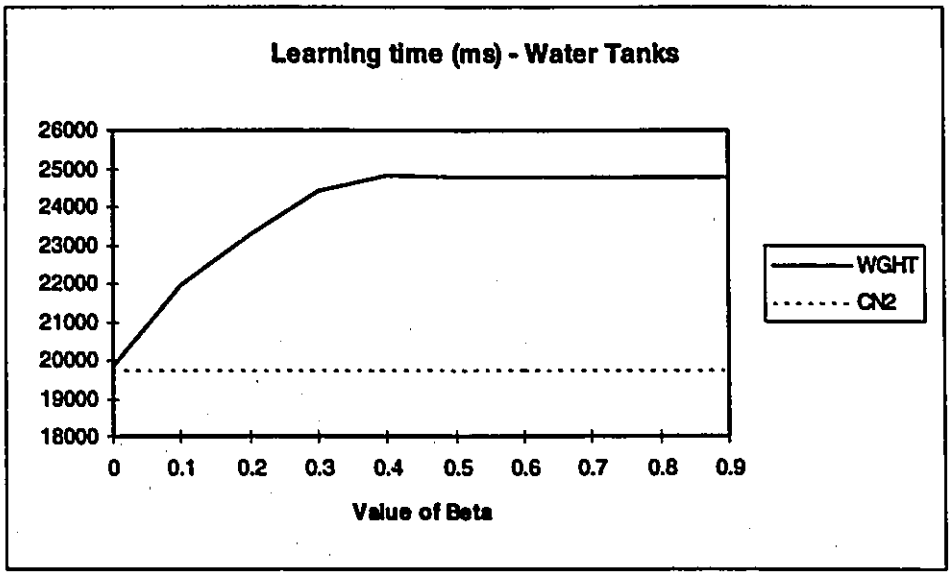


Figure 6.17

increases with β. The more belief values of rules are taken into account in the search heuristic, the less often non-explainable rules are selected. Note that even for β=0.1, the explainability of the learned ruleset for the three domains is higher than the explainability of CN2's results by at least 30 %. For example, in the water tanks domain, with β set to 0.2, WGHT learned rules that were approximately 1% more accurate and 50% more explainable with respect to the provided qualitative model. On the other hand, the advantage of using WGHT rather than TCS when the model is good is not obvious. The small gain in accuracy is lost in explainability. Since TCS's computational cost is lower than WGHT's, it is probably advantageous to use TCS rather than WGHT, at least for applications where a small lost of classificational accuracy is not important compared to the advantage of a total explainability with respect to the qualitative model.

Surprisingly, as shown in Figure 6.17, the learning time increases with the value of β. Even when only explainable rules are selected, as shown by the total explainability of the results for high values of β, the learning time is much higher than when belief weights are not taken into account, i.e. than when WGHT behaves like CN2. We believe that it illustrates that it is easier for the inductive learner to obtain a satisfying coverage of the positive training examples when all the possible rules are available rather than only explainable rules. The size of the learned rulesets (measured in terms of the number of parameter tests) was 20 % higher for high values of β than for β set to 0.0, and thus indicates that either the learner had to produce a larger number or more specialised rules to obtain an equivalent coverage of the positives examples, resulting in more work.

In order to determine how well WGHT compensates for incomplete knowledge by allowing non-explainable rules to be selected, an experiment similar to WT1 (see previous section) was conducted. However, rather than removing the I*+ link connecting L6 to L7 in the qualitative model of the water tank system, the value of the associated belief weight was made to vary between 0.0 and 0.8. Note that since by convention, links which are not in the qualitative model are considered to have a belief weight equal to 0.0. Consequently,

setting the belief weight of the link of interest to 0.0 has the same effect as actually removing it from the model.

For this experiment, the depth limit was set to 3, the beam width was set to 5, and 65% of the whole data set associated with the water tanks system was used for training. Figure 6.18 and Figure 6.19 show respectively the measured accuracy and explainability of the learned rulesets. The horizontal axis indicates the value of β, like in the previous graphs, and each curve corresponds to a given value of the belief weight associated with the link of interest in the qualitative model.

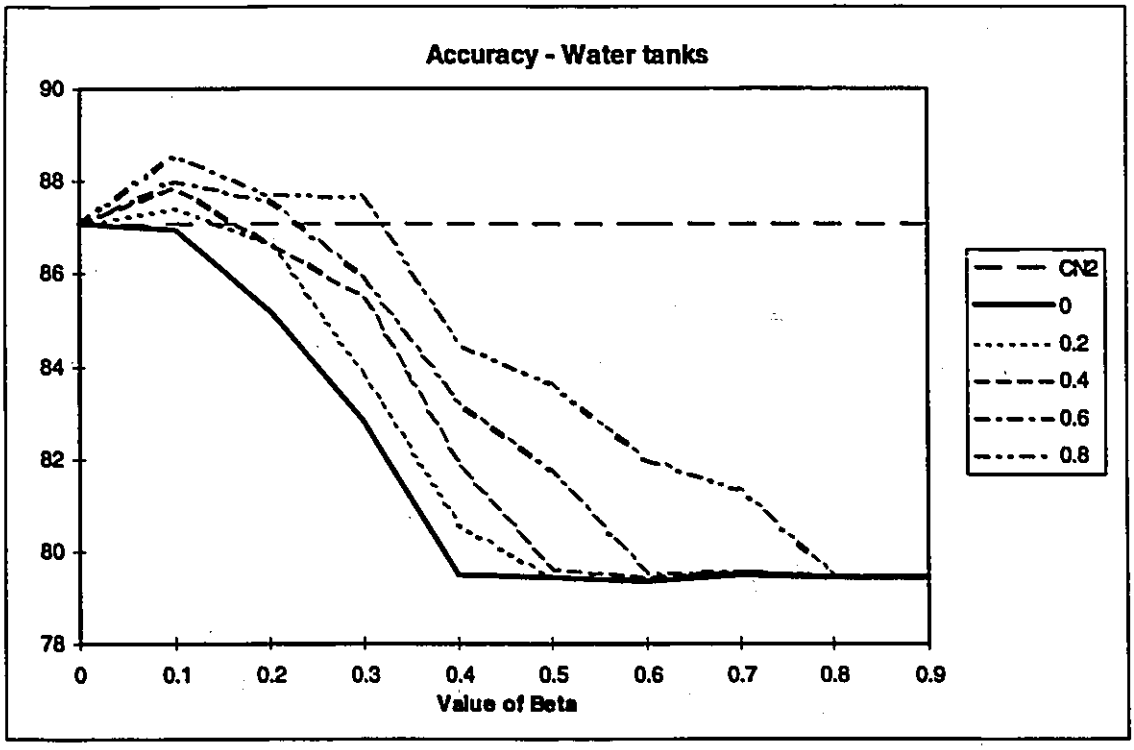


Figure 6.18

To allow comparisons, the accuracy and the explainability of the rulesets induced by CN2, both independent of the values of the belief weight value and of β, were added on the two charts.

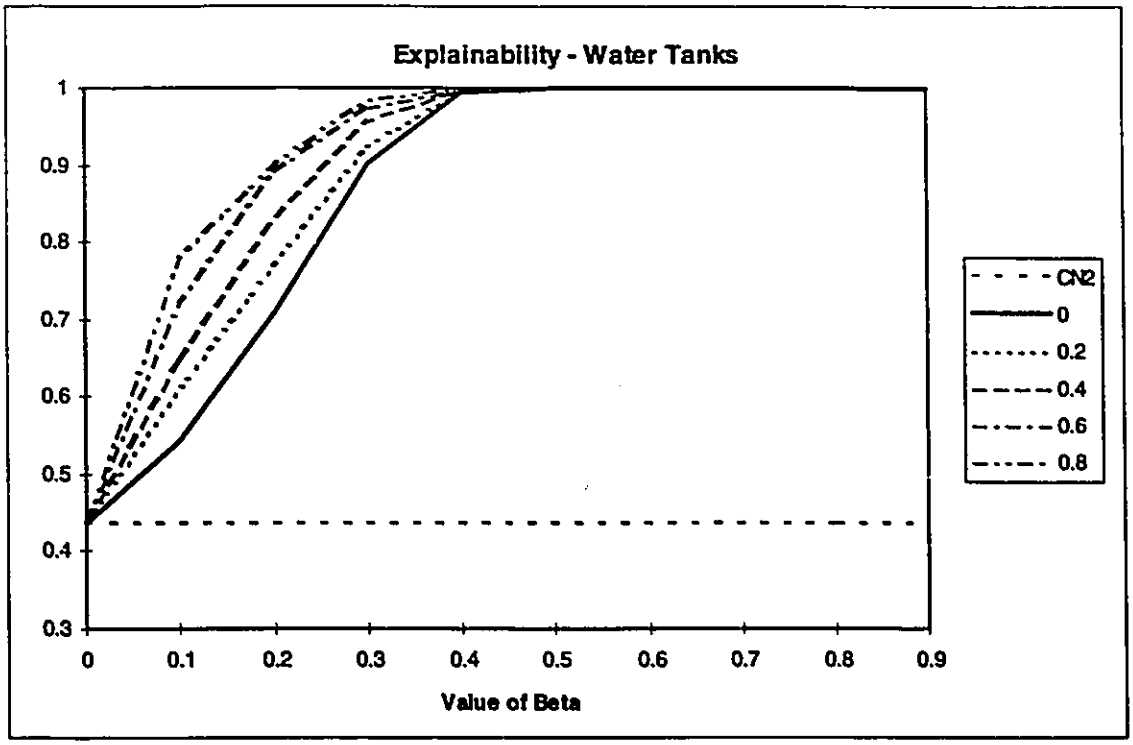


Figure 6.19

These experimental results show that the value of the belief weight associated with the link of interest has indeed a strong impact on the accuracy and explainability of the rulesets produced by WGHT. When this belief weight is equal to 0 (it is as if the link was removed from the model), rules accuracy deteriorates seriously for high values of β and illustrates the fact that the qualitative model is incomplete. Furthermore, both accuracy and explainability improve uniformly when the value of the belief weight increases. It denotes that the higher the belief weight of the link of interest, the more often rules explained by this link are induced. This gives a 'bonus' to rules that involve reliable parts of the models, as specified by the setting of belief weights in the model by the expert. These rules are likely to be more accurate than rules involving lower belief weights, and that are considered less reliable by the expert. Note that another way to interpret WGHT's behavior is to consider that the model's incompleteness actually decreases when the belief weight associated with the link of interest increases.

The conclusion we derive from this experiment is that, if a link is important, in the sense that it is consistent with the actual physical system and useful for the learning task, the results obtained with WGHT improves uniformly with the belief weight associated with this link. Our experiment shows also that, for low values of β, a belief weight of 0.2 already improves accuracy over the standard CN2.

Furthermore, for any value of the belief weight, it is possible to obtain a trade-off between the high accuracy of rules induced by CN2 and the total explainability of rules produced by TCS, by choosing adequately the value of the parameter β. For low values of β, rules will have a high accuracy but a low explainability. For high values of β, rules may have a lower explainability if the model is incomplete but, on the other hand, have a high explainability.

Chapter 7

Conclusion

In this last chapter, we present our conclusions about the work presented in this thesis. We also discuss possible extensions to this research.

7.1 Conclusion

In this thesis, we have presented our research on the application of qualitative knowledge to guide inductive learning. In our particular implementation, the background knowledge is a qualitative model of the physical system under study and the inductive algorithm is CN2. We have defined the notion of explainability of a rule with respect to a qualitative model in order to be able to distinguish rules which are consistent with our knowledge of the physical system from those which contradicts this knowledge. Additionally, we have presented three modifications of the standard CN2 algorithm which take into account in various ways this notion of explainability during learning.

In doing this, we had several objectives:

• our first objective was to improve the interpretability of the learned rules. When a rule is explainable with respect to a qualitative model, it becomes possible to derive a qualitative and causal explanation of this rule. We believe that this qualitative explanation considerably improves the comprehensibility of the results of learning. Such a comprehensibility is essential for both practical applications of machine learning and for possible integration of the results of learning into an existing knowledge base.

• our second objective was to try to improve the classificational accuracy of induced rules by using the background knowledge to guide the inductive hill-climbing search. By biasing the search in favor of explainable rules, we expected to prune out rules which by chance perform well on the training examples, but which will have a poor accuracy on unseen examples.

• the qualitative model may not be perfect. Our third objective, and the main contribution of this thesis, consisted in developing learning algorithms which are able to compensate for possible imperfections of the qualitative model by allowing a trade-off between explainability and accuracy of the induced ruleset. Although it was not one of our main objectives, we also wanted to find a way to try to revise an imperfect model according to evidence found in the training data.

Furthermore, we wanted to ease and improve the design of qualitative models. We thus extended the representation language of our qualitative models to allow for the representation of uncertainty.

In response to these objectives, experimental evaluation of the methods we propose showed that:

• qualitative knowledge can successfully be applied to guide qualitative learning. Provided with a reasonably correct and complete qualitative model, our first approach, the totally constrained search (TCS) obtains totally explainable rules at least as accurate as rules induced by CN2. For some of the application domains, the accuracy was even significantly improved. Another benefit of using this first approach is a considerable reduction off the learning time.

• when the available qualitative model is incomplete, the totally constrained search algorithm produces rules with lower accuracy than rules induced by CN2. On the other hand, CN2's results are often poorly explainable. We developed two methods (THR and WGHT) which successfully allow a trade-off between CN2's high accuracy and the total

explainability obtained by the totally constrained search algorithm (TCS). This trade-off allows an optimal exploitation of the qualitative model to produce rules with a high accuracy and a high explainability.

• The threshold algorithm (THR) that we developed can be used to suggest completion of an incomplete qualitative model. While these completions are limited in the sense that they do not provide information about the internal structure of the physical system, they give good evidence of the influence of the system's parameters on the target parameter.

• our learning algorithm, which takes uncertainty in the qualitative model into account during learning (WGHT), often improves classificational accuracy over CN2. Furthermore, when a link is important, WGHT's performance is consistently and uniformly improved with the belief weight associated to this link. It thus shows that WGHT efficiently exploits the measure of uncertainty introduced in this work.

7.2 Future work and possible extensions

While an already important part of this thesis is devoted to the experimental evaluation of the effects of the imperfectness of a qualitative model on learning performance, we did not consider the case where model was not incomplete but incorrect. The study of the effects of the correctness of the model on learning, and of the possible corrections which could be brought up in case the model is incorrect can thus be an interesting extension of our work.

Also, we defined the explainability of a rule in a Boolean manner: a rule is either explainable or non-explainable. While this definition is the translation of our ability (or of our inability) to derive a qualitative explanation of the rule from the qualitative model, we can also consider alternatives. A possible approach could be to define the explainability of a rule as the percentage of its attribute tests which can be explained, with respect to the fully specified rules extracted from the qualitative model. Furthermore, this new measure of explainability can also be used as a search heuristic: we can impose a minimum

explainability for the rules to be induced. This would actually be an extension of TCS, for which the constant explainability is 1. Note that the introduction of this new measure of explainability would change some aspects of our research. For example, for THR, it would involve a change in the learning task. Rather than trying to maximize the number of totally explainable rules in the ruleset, while also maintaining accuracy with comparison to CN2, the goal would become to try to maximize the proportion of explainable parameter tests in the ruleset. As a consequence, one might consider the interest of trying to 'come back' to the qualitative model after a search expansion. In our current implementation, once a search expansion has occurred, we keep on specialising by adding the best parameter test, explainable or not. Instead, after a search expansion, it would probably be a best choice (from the point of view of the new measure of explainability) to try to restrict again the specialisation operation to explainable parameter tests[1].

Finally, WGHT's performance was observed to improve monotonically with the belief weight associated with a pertinent link. If WGHT, as perhaps proven by further experiments, sees a decrease in its performance with the belief weight of an incorrect link, an exciting possibility of using WGHT to perform an hill-climbing search for an optimal qualitative model, given a set of training examples, is attainable. THR's ability for completion could also interestingly be exploited to build a first approximation of a qualitative model from scratch. This first draft could then be refined and optimized by hill-climbing search based on WGHT's performance, in the sense that WGHT, being more sensible than THR to the particular value of each link, would indicate the important links as discussed in section 6.2.3.

[1] Note that with our Boolean measure of explainability, it does not really make sense to try to add only explainable parameter test again, once a non-explainable specialisation has made the rule non-explainable.

References

[Bergadano and Giordana, 1988] Bergadano, F. and Giordana, A. A knowledge intensive approach to concept induction. In Laird, J., editor, *ML-88 (Proc. Fifth Int. Machine Learning Conference)*, pp305-317, Ca. Kaufmann.

[Bratko et al., 1989] Bratko, I., Mozetic, I., and Lavrac, N. *Kardio: A Study in Deep and Qualitative Knowledge for Expert Systems.* MIT Press, Cambridge, Ma.

[Cestnik, 1990] Cestnik, B. Estimating probabilities: A crucial task in machine learning. In *ECAI-90.*

[Cestnik and Bratko, 1991] Cestnik, B. and Bratko, I. On estimating probabilities in tree pruning. In Kodratoff, Y., editor, *Proc. EWSL-91.*

[Clark and Boswell, 1991] Clark, P. and Boswell, R. Rule induction with CN2: Some recent improvements. In Kodratoff, Y, editor, *Machine Learning - EWSL-91*, pp151-163, Berlin. Springer-Verlag.

[Clark and Matwin, 1993] Clark, P. and Matwin, S. Using qualitative models to guide inductive learning. In *Proceedings of the Tenth International Conference on Machine Learning*, pp49-56, Amherst, MA.

[Clark and Niblett, 1989] Clark, P. and Niblett, T. The CN2 induction algorithm. *Machine Learning Journal*, 3(4), pp261-283.

[Efron and Gong, 1983] Efron, B. and Gong, G. A leisurely look at the bootstrap, the jackknife, and cross validation. *The American Statistician*, 37(1), pp36-48.

[Evans and Fisher, 1993] Evans, R. R. and Fisher, D. Overcoming process delays with decision tree induction. *IEEE Expert*, 9(1), pp60-66.

[Flann and Dietterich, 1989] Flann, N. and Dietterich, T. A study of explanation based methods for inductive learning. *Machine Learning*, 4, pp187-226.

[Forbus, 1984] Forbus, K. D. Qualitative process theory. *Artificial Intelligence*, 24, pp85-168.

[JKTech Ltd., 1991] *The JKSimMet Steady State Mineral Processing Simulator*. JKTech, QLD, Australia.

[Marchand, 1995] Marchand. Industrial Experiences in Comprehensibility. *IJCAI'95 Workshop in Machine Learning and Comprehensability*, pp125-131.

[Ortega and Fisher, 1995] Ortega, J. and Fisher, D. Flexibly exploiting prior knowledge in empirical learning. *International Joint Conferences on Artificial Intelligence* 1995 (IJCAII).

[Ourston and Mooney, 1994] Ourston, D. and Mooney, R. Chaining the rules: A comprehensive approach to theory refinement. *Proceedings of the Eighth National Conference on Artificial Intelligence*, pp815-820.

[Pazzani and Kibler, 1992] Pazzani, M. and Kibler, D. The utility of knowledge in inductive learning. *Machine Learning*, 9, pp57-94.

[Quinlan, 1990] Quinlan, J. R. Learning logical definitions from relations. *Machine Learning*, 5, pp239-266.

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

Appendix A

Example of specialisation lattice file

In this section, we present an example of a lattice file, generated from the analog electric circuit domain. This corresponds to the actual Prolog clauses as they are loaded before learning.

The qualitative model file is:

```
:- op(910, fx, link).
:- op(900, xfy, [
        '-- Q+ -->', '-- Q- -->',
        '-- I+ -->', '-- I- -->',
        '-- I*+ -->', '-- I*- -->']).

observables([p1,p2,p3,p4,p5,p6,p8,p10,p11]).

target(p11).

link    p1 '-- Q- -->' p5.
link    p2 '-- Q- -->' p5.
link    p3 '-- I*- -->' p6.
link    p4 '-- I*- -->' p7.
link    p5 '-- Q- -->' p8.
link    p6 '-- Q- -->' p8.
link    p7 '-- Q+ -->' p10.
link    p8 '-- I*- -->' p9.
link    p9 '-- Q+ -->' p10.
link    p9 '-- Q- -->' p5.
link    p10 '-- I*- -->' p11.

text(p1, 'the voltage at node 1').
text(p2, 'the voltage at node 2').
text(p3, 'the voltage at node 3').
text(p4, 'the voltage at node 4').
text(p5, 'the voltage at node 5').
text(p6, 'the voltage at node 6').
text(p7, 'the voltage at node 7').
text(p8, 'the voltage at node 8').
text(p9, 'the voltage at node 9').
text(p10, 'the voltage at node 10').
```

Corresponding generated lattice file:

/* First are included the extracted fully specified rules */

/* rule(Rule_number, rule([CauseEvidencePath], [PreEffectEvidencePath] [PropagationPath]) */

```
rule(1,rule([[p3/ (+)]],[[p6/ (+)]],[p11/ (+),istar,p10/ (-),q,p9/ (-),istar,p8/ (+),q,p6/
(-),istar,p3/ (+)])).
rule(2,rule([[p4/ (+)]],[[p10/ (+),q,p7/ (+)]],[p11/ (+),istar,p10/ (-),q,p7/ (-
),istar,p4/ (+)])).
rule(3,rule([[p8/ (+)]],[[p5/ (-),q,p9/ (+)],[p10/ (+),q,p9/ (+)]],[p11/ (+),istar,p10/ (-
),q,p9/ (-),istar,p8/ (+)])).
rule(4,rule([[p10/ (-)]],[[p11/ (-)]],[p11/ (+),istar,p10/ (-)])).
```

/* Here starts the specialisation lattice */

/* qm_specialisations(HashIndex, Class, CurrentRule, FulSpeRulNum, ExplainSpec) */

/* HashIndex: hash version of CurrentRule */

/* Class: class to predict */

/* CurrentRule: list of the parameter tests of the rule to specialise */

/* FulSpeRulNum: Number of fully specified rules which explain the current rule */

/* ExplainSpec: explainable specialisations for CurrentRule */

```
:- dynamic qm_specialisations/5.

qm_specialisations(root,increase,root,[],[p10/[<],p3/[>],p4/[>],p8/[>]]).
qm_specialisations(root,decrease,root,[],[p10/[>],p3/[<],p4/[<],p8/[<]]).
qm_specialisations(p8,increase,3,[p8/ >],[p10/[<,>],p3/[>],p4/[>],p5/[<]]).
qm_specialisations(p8,decrease,3,[p8/ <],[p10/[<,>],p3/[<],p4/[<],p5/[>]]).
qm_specialisations('p5<p8>',increase,3,[p5/ <,p8/ >],[p10/[<,>],p3/[>],p4/[>]]).
qm_specialisations('p5>p8<',decrease,3,[p5/ >,p8/ <],[p10/[<,>],p3/[<],p4/[<]]).
qm_specialisations('p4>p8>',increase,[2,3],[p4/ >,p8/ >],[p10/[>],p5/[<]]).
qm_specialisations('p4<p8<',decrease,[2,3],[p4/ <,p8/ <],[p10/[<],p5/[>]]).
qm_specialisations('p4>p5<p8>',increase,[2,3],[p4/ >,p5/ <,p8/ >],[p10/[>]]).
qm_specialisations('p4<p5>p8<',decrease,[2,3],[p4/ <,p5/ >,p8/ <],[p10/[<]]).
qm_specialisations(p4,increase,2,[p4/ >],[p10/[<,>],p3/[>],p8/[>]]).
qm_specialisations(p4,decrease,2,[p4/ <],[p10/[<,>],p3/[<],p8/[<]]).
qm_specialisations('p3>p8>',increase,[1,3],[p3/ >,p8/ >],[p10/[>],p5/[<],p6/[>]]).
qm_specialisations('p3<p8<',decrease,[1,3],[p3/ <,p8/ <],[p10/[<],p5/[>],p6/[<]]).
qm_specialisations('p3>p6>p8>',increase,[1,3],[p3/ >,p6/ >,p8/ >],[p10/[>],p5/[<]]).
qm_specialisations('p3<p6<p8<',decrease,[1,3],[p3/ <,p6/ <,p8/ <],[p10/[<],p5/[>]]).
qm_specialisations('p3>p6>',increase,1,[p3/ >,p6/ >],[p10/[<],p4/[>],p8/[>]]).
qm_specialisations('p3<p6<',decrease,1,[p3/ <,p6/ <],[p10/[>],p4/[<],p8/[<]]).
qm_specialisations('p3>p5<p8>',increase,[1,3],[p3/ >,p5/ <,p8/ >],[p10/[>],p6/[>]]).
qm_specialisations('p3<p5>p8<',decrease,[1,3],[p3/ <,p5/ >,p8/ <],[p10/[<],p6/[<]]).
qm_specialisations('p3>p5<p6>p8>',increase,[1,3],[p3/ >,p5/ <,p6/ >,p8/ >],[p10/[>]]).
qm_specialisations('p3<p5>p6<p8<',decrease,[1,3],[p3/ <,p5/ >,p6/ <,p8/ <],[p10/[<]]).
qm_specialisations('p3>p4>p6>',increase,[1,2],[p3/ >,p4/ >,p6/ >],[p10/[>]]).
qm_specialisations('p3<p4<p6<',decrease,[1,2],[p3/ <,p4/ <,p6/ <],[p10/[<]]).
qm_specialisations('p3>p4>',increase,[1,2],[p3/ >,p4/ >],[p10/[>],p6/[>]]).
qm_specialisations('p3<p4<',decrease,[1,2],[p3/ <,p4/ <],[p10/[<],p6/[<]]).
qm_specialisations(p3,increase,1,[p3/ >],[p10/[<],p4/[>],p6/[>],p8/[>]]).
qm_specialisations(p3,decrease,1,[p3/ <],[p10/[>],p4/[<],p6/[<],p8/[<]]).
qm_specialisations('p10>p8>',increase,3,[p10/ >,p8/ >],[p3/[>],p4/[>],p5/[<]]).
qm_specialisations('p10<p8>',increase,[3,4],[p10/ <,p8/ >],[p11/[<],p5/[<]]).
qm_specialisations('p10<p8<',decrease,3,[p10/ <,p8/ <],[p3/[<],p4/[<],p5/[>]]).
qm_specialisations('p10>p8<',decrease,[3,4],[p10/ >,p8/ <],[p11/[>],p5/[>]]).
qm_specialisations('p10>p5<p8>',increase,3,[p10/ >,p5/ <,p8/ >],[p3/[>],p4/[>]]).
qm_specialisations('p10<p5<p8>',increase,[3,4],[p10/ <,p5/ <,p8/ >],[p11/[<]]).
qm_specialisations('p10<p5>p8<',decrease,3,[p10/ <,p5/ >,p8/ <],[p3/[<],p4/[<]]).
qm_specialisations('p10>p5>p8<',decrease,[3,4],[p10/ >,p5/ >,p8/ <],[p11/[>]]).
qm_specialisations('p10>p4>p8>',increase,[2,3],[p10/ >,p4/ >,p8/ >],[p5/[<]]).
qm_specialisations('p10<p4<p8<',decrease,[2,3],[p10/ <,p4/ <,p8/ <],[p5/[>]]).
qm_specialisations('p10>p4>p5<p8>',increase,[2,3],[p10/ >,p4/ >,p5/ <,p8/ >],[]).
qm_specialisations('p10<p4<p5>p8<',decrease,[2,3],[p10/ <,p4/ <,p5/ >,p8/ <],[]).
```

```
qm_specialisations('p10>p4>',increase,2,[p10/ >,p4/ >],[p3/[>],p8/[>]]).
qm_specialisations('p10<p4>',increase,[2,4],[p10/ <,p4/ >],[p11/[<]]).
qm_specialisations('p10<p4<',decrease,2,[p10/ <,p4/ <],[p3/[<],p8/[<]]).
qm_specialisations('p10>p4<',decrease,[2,4],[p10/ >,p4/ <],[p11/[>]]).
qm_specialisations('p10>p3>p8>',increase,[1,3],[p10/ >,p3/ >,p8/ >],[p5/[<],p6/[>]]).
qm_specialisations('p10<p3<p8<',decrease,[1,3],[p10/ <,p3/ <,p8/ <],[p5/[>],p6/[<]]).
qm_specialisations('p10>p3>p6>p8>',increase,[1,3],[p10/ >,p3/ >,p6/ >,p8/ >],[p5/[<]]).
qm_specialisations('p10<p3<p6<p8<',decrease,[1,3],[p10/ <,p3/ <,p6/ <,p8/ <],[p5/[>]]).
qm_specialisations('p10>p3>p6>',increase,[1,4],[p10/ <,p3/ >,p6/ >],[p11/[<]]).
qm_specialisations('p10>p3<p6<',decrease,[1,4],[p10/ >,p3/ <,p6/ <],[p11/[>]]).
qm_specialisations('p10>p3>p5<p8>',increase,[1,3],[p10/ >,p3/ >,p5/ <,p8/ >],[p6/[>]]).
qm_specialisations('p10<p3<p5>p8<',decrease,[1,3],[p10/ <,p3/ <,p5/ >,p8/ <],[p6/[<]]).
qm_specialisations('p10>p3>p5<p6>',increase,[1,3],[p10/ >,p3/ >,p5/ <,p6/ >,p8/ >],[]).
qm_specialisations('p10<p3<p5>p6<',decrease,[1,3],[p10/ <,p3/ <,p5/ >,p6/ <,p8/ <],[]).
qm_specialisations('p10>p3>p4>p6>',increase,[1,2],[p10/ >,p3/ >,p4/ >,p6/ >],[]).
qm_specialisations('p10<p3<p4<p6<',decrease,[1,2],[p10/ <,p3/ <,p4/ <,p6/ <],[]).
qm_specialisations('p10>p3>p4>',increase,[1,2],[p10/ >,p3/ >,p4/ >],[p6/[>]]).
qm_specialisations('p10<p3<p4<',decrease,[1,2],[p10/ <,p3/ <,p4/ <],[p6/[<]]).
qm_specialisations('p10<p3>',increase,[1,4],[p10/ <,p3/ >],[p11/[<],p6/[>]]).
qm_specialisations('p10>p3<',decrease,[1,4],[p10/ >,p3/ <],[p11/[>],p6/[<]]).
qm_specialisations('p10<p11<p8>',increase,[3,4],[p10/ <,p11/ <,p8/ >],[p5/[<]]).
qm_specialisations('p10>p11>p8<',decrease,[3,4],[p10/ >,p11/ >,p8/ <],[p5/[>]]).
qm_specialisations('p10<p11<p5<p8>',increase,[3,4],[p10/ <,p11/ <,p5/ <,p8/ >],[]).
qm_specialisations('p10>p11>p5>p8<',decrease,[3,4],[p10/ >,p11/ >,p5/ >,p8/ <],[]).
qm_specialisations('p10<p11<p4>',increase,[2,4],[p10/ <,p11/ <,p4/ >],[]).
qm_specialisations('p10>p11>p4<',decrease,[2,4],[p10/ >,p11/ >,p4/ <],[]).
qm_specialisations('p10<p11<p3>p6>',increase,[1,4],[p10/ <,p11/ <,p3/ >,p6/ >],[]).
qm_specialisations('p10>p11>p3<p6<',decrease,[1,4],[p10/ >,p11/ >,p3/ <,p6/ <],[]).
qm_specialisations('p10<p11<p3>',increase,[1,4],[p10/ <,p11/ <,p3/ >],[p6/[>]]).
qm_specialisations('p10>p11>p3<',decrease,[1,4],[p10/ >,p11/ >,p3/ <],[p6/[<]]).
qm_specialisations('p10<p11<',increase,4,[p10/ <,p11/ <],[p3/[>],p4/[>],p8/[>]]).
qm_specialisations('p10>p11>',decrease,4,[p10/ >,p11/ >],[p3/[<],p4/[<],p8/[<]]).
qm_specialisations(p10,increase,4,[p10/ <],[p11/[<],p3/[>],p4/[>],p8/[>]]).
qm_specialisations(p10,decrease,4,[p10/ >],[p11/[>],p3/[<],p4/[<],p8/[<]]).
```