



Norwegian University of
Science and Technology

Explanation Methods in Clinical Decision Support

A Hybrid System Approach

Kim Ohme Pedersen

Master of Science in Computer Science

Submission date: June 2010

Supervisor: Agnar Aamodt, IDI

Co-supervisor: Helge Langseth, IDI
Tore Bruland, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Problem Description

In a cooperation project with the Palliative Care Unit at St. Olavs Hospital, we are studying the use of case-based reasoning (CBR) and Bayesian networks (BN) to support clinical personnel in the classification and treatment of cancer pain. Our activities are coupled to ongoing EU and NFR projects, in particular the NFR funded project TLCPC.

In this Master Project the problem of explaining the system's suggestions, in order to increase the clinician's understanding and trust, will be studied. The underlying problem solving method will combine case-based and Bayesian reasoning, and the explanation method need to reflect that. A method shall be designed that identify one or two explanation goals, based on a pre-study made in the earlier specialization project, for example transparency and justification of the system's behaviour. An experimental implementation that demonstrate the explanation method will be made, based on the existing CBR and BN tools within the TLCPC research group, i.e. jColibri and Smile/Genie, respectively.

Application of the method to case data and Bayesian models for palliative pain requires that these data and models become available in due time, as is the plan. In case of delays, and intermediate or different application will be selected, which is similar enough so that essential properties of clinical decision making is kept.

The project work will be done in close contact with the TLCPC project group at IDI and St. Olavs Hospital.

Assignment given: 22. January 2010
Supervisor: Agnar Aamodt, IDI

Abstract

The use of computer-based decision support systems within the field of health science has over the last decades been extensively researched and tested, both in controlled environments and in clinical practice. Despite the obvious benefits of utilising such systems in day-to-day activities, many of the designed systems fail to make the impact one could hope to achieve. We have designed and implemented a prototype of a decision support system which use both Case-Based Reasoning and probabilistic inference through a Bayesian Network as a basis for the solution. To achieve user acceptance an explanation module has been implemented which gives the user full access to the data which has been used in the reasoning process, both from the Case-Based Reasoning and the Bayesian Network. The system has shown promising results within the domain of wine recommendation, with a very high accuracy despite uncertain precision of the knowledge within the system. Furthermore the explanations presented to an expert conformed to the causal way of reasoning used by said expert, and was accepted as a very useful tool to get pointed in the right direction for an evaluation of the solution.

Preface

This report documents my achievements of the 10th and final semester of the Master of Science studies in Computer Science at the Norwegian University of Science and Technology (NTNU). It was carried out at the Department of Computer and Information Science (IDI). The work started 22nd January, 2010 and ended 24th June, 2010.

First a thank to my college and co-developer, Anne-Marit Gravem, is in order. Without our discussions the result would be all the poorer. The initiative to the thesis was Agnar Aamodt's, and I would like to thank him for the invaluable advice and the ability to point me in the right direction whenever stuck. I would also like to thank the two bi-supervisors, Tore Bruland and Helge Langseth, which has given us new suggestions and help when we needed it most. Lastly I would like to thank Jorunn Hoøen from Vinmonopolet for her help with testing and evaluation of the results from the system.

Trondheim, 24th June, 2010.

Kim Ohme Pedersen

Contents

1	Introduction	1
1.1	Introduction	1
1.2	Motivation	2
1.3	Goals	3
1.4	Scope	4
1.5	Methodology	4
1.6	Co-operation Based Implementation	5
1.7	Previous Work	6
2	Background	9
2.1	The TLCPC Project	9
2.2	Hypothesis	10
2.3	Case-Based Reasoning	11
2.4	Bayesian Network	13
2.5	Related Work	14
3	Approaches to Explanation	18
3.1	The Explanation Goals	19
3.2	Case-Based Reasoning and Explanation	21
3.3	Explanations in Bayesian Networks	23
4	System Solution	29
4.1	The Domain	29
4.2	Design Overview	30
4.3	Cases	33
4.4	The Case-base	35
4.5	The Case-Based Reasoning Part	35
4.6	The Bayesian Network Part	37
4.7	The Graphical User Interface	41

4.8	Running Bacchus	43
5	Implementation	50
5.1	Technology	50
5.1.1	jColibri	50
5.1.2	GeNIe / SMILE	51
5.1.3	Eclipse	51
5.1.4	Apache Derby	52
5.1.5	Hibernate	52
5.1.6	Protegé / myCBR	52
5.2	Implementational details	52
5.3	The Explanation Generation	56
6	Results and Discussion	58
6.1	Results	59
6.1.1	Discussion of the Progress	67
6.2	Future Work	68
7	Conclusion	71
	References	72

Chapter 1

Introduction

1.1 Introduction

The use of computer-based decision support systems within the field of health science has over the last decades been extensively researched and tested, both in controlled environments and in clinical practice. Despite the obvious benefits of utilising such systems in day-to-day activities, many of the designed systems fail to make the impact one could hope to achieve.

Within the field of artificial intelligence a lot of effort has been invested in how to make software able to adapt to new situations and, perhaps specifically for the sub-field of machine learning, learn from past experiences. Both of these qualities are invaluable in highly dynamic environments where new situations occur daily and new experiences are made.

Currently, a research team consisting of members from the Department of Computer and Information Science (IDI) and from St.Olavs Hospital in Trondheim is working on a computer based tool to help medical professionals come up with the best possible treatment in palliative care. The system will make use of a combination of two machine learning paradigms: Case-Based Reasoning (CBR) and Bayesian Networks (BN). The combination of these two makes for an interesting synergy between concrete cases and collected statistical data, which together provides a strongly evidence based solution.

This thesis will explore how to explain the results presented to the user by such a system. Explanations of case-solutions in e.g. Case-Based Explanation systems has been studied by researchers for some time now, but as far as we know there has been no earlier attempts to combine previous experiences stored in cases and

statistical data into coherent explanations within health science. In fall 2009 the author did a specialisation project related to this subject where the goal was to design a proposal for how to generate explanations in such a hybrid system. A brief overview of the project and the results found there will be presented in Section 1.7.

The foundation for the thesis is based upon the thought that there has been no attempt at combining these paradigms before. The goal is to design a prototypical system that makes use of the strengths of both CBR and BN and from there generate explanations that combine the best of these two paradigms. The goal and scope of this thesis will be further specified in sections 1.3 and 1.4.

The report is structured as follows. Chapter 1 gives the reader an introduction to the report and the motivation behind it, the scope and the goal, as well as the methodology used to achieve this goal, is also presented here, while a brief overview of previous work done by the author is presented last. In Chapter 2 background information about the project and the methods used is given, ending by a look at similar systems already developed or currently being worked on. A presentation of common approaches to explanations within the field is presented in Chapter 3. The system design and solution is gone through in Chapter 4, while Chapter 5 gives the implementational details behind the system. In Chapter 6 the results of the project along with a discussion of these and future work is given, while the last Chapter, 7, concludes the work done.

1.2 Motivation

The motivation for the project, and thereby this thesis, is twofold. One of the motivational points is related to the medical science part and the TLCPC project. The second part is tied to the field of computer science, where research on reasoning systems within the machine-learning field is the motivation.

In palliative care the goal is to relieve the pain of cancer patients and provide them with a comfortable life as far as possible into the illness. Cancer is a disease that can be caused by several factors, like e.g. smoking or, as growing evidence shows, overweight (Bergström et al., 2001). In addition there is a *possibility* for some people being more susceptible to certain kinds of cancer because of heritable genetics (Lichtenstein et al., 2000), and thus it affects thousands of people here in Norway and many more world-wide. The importance of giving all these people an as optimal treatment as possible is therefore quite a noble task, and one which will be appreciated by countless families.

Research on artificial intelligence has been going on for several decades already,

with focus on everything from chess-playing machines to “robots” to intelligent search engines. In (Lacave and Díez J, 2002) the authors quote William J. Clancey saying:

The key issue is not *artificial* intelligence, but how to extend *natural* intelligence through knowledge-based systems.

The author of this report has not been able to confirm whether or not the quote is actual, but that is besides the point. The point is that this particular quote nicely sums up how the system that has been designed in relation to this project is supposed to work: As a support, extension and helper system for human beings doing their job to make them even better at it. A whole field of research has been centered around this way of seeing artificial intelligence, and the result, among other things, has been the design of several (maybe hundreds in the number) *Decision Support Systems* (DSS). As the name implies these are systems made in order to guide and support humans in the making of decisions. Within health science a whole branch of these kinds of systems are specialised in decision support within the field, and has thus been labelled *Clinical DSS*.

As motivation, the fact that so many CDSS systems are already being used shows that there is a need for these kinds of applications. Furthermore, since the research has only been going on for a few decades there are probably a lot of holes to fill. Not only that, but there is always room for improvement and that is the aim for this project. To build a system that combines two paradigms already proven successful in CDSS in order to make the reasoning process even better and more reliable.

As for this thesis' part in the overall project, the motivation is to help make this system more helpful by providing explanations for how and possibly why the system reached the conclusion it did.

1.3 Goals

The goal of the thesis is to provide a prototype implementation of a system to show how explanations can be generated from a combination of CBR and BN, in what degree these will be useful compared to other methods and in what way these can be improved upon. Furthermore, the report will through literature and testing of the prototype give a thorough understanding of how a combination of explanations from the two paradigms can improve the acceptance and usability of a reasoning system.

1.4 Scope

The thesis will include the implementation of a prototype system with basic Case-Based Reasoning and Bayesian Network functionality. The CBR system will be limited to the first step of the cycle, retrieval, and will thus not implement functionality of reuse, revise or retain. The BN will be a fully functional network structure with Bayesian probability tables, but will not support learning from or of new data.

In addition to the system, the thesis will provide a prototype implementation of explanation mechanics. Three different sorts will be implemented, a case-based explanation, a bayesian network explanation and a combination of the two, with emphasis on the latter.

Furthermore, the report will give a thorough discussion of the implementation and the pros and cons of the implementation. There will also be an evaluation of the results found by the explanation mechanism implementation, but no in-depth discussion of the system implementation itself, i.e., the combination of CBR and BN.

1.5 Methodology

Given the wide array of research done on the subjects of case-based reasoning, bayesian networks and explanations, the first phase of the project was dedicated to a pre-study. Since the specialisation project done last fall (see Section 1.7) was a literature study on the same subject, time was spent on re-familiarising with the sources used there as well as the information in the report from that project. In addition, new sources of information has been perused in order to complement and elaborate on information not studied in the previous work. Most prominently this involved a study of Bayesian networks and explanations in relation to probabilistic data, but also various case-based explanation methods. An overview of methods within these areas is presented in Chapter 3.

The use of frameworks for implementation of both the CBR and the BN part of the prototype system in this project demanded time spent to understand and learn the particular techniques.

Analysis of the system design was needed in order to assess what kind of information would be available for use as explanation basis. Ideally this should happen the other way around, i.e., analysis of the wanted explanation capabilities should guide

the implementation. However, since the system has been developed as a co-effort, and because of the use of frameworks in the implementation phase, this was not really a possibility.

Implementation of the prototype system has been a large part of the project and was a cooperation between Anne-Marit Gravem and me. The particulars of this work is discussed in Section 1.6.

The system and the explanations produced by it has been tested against an expert within the chosen domain. The results of these testings are given in Chapter 6.

1.6 Co-operation Based Implementation

The prototype system implemented for this project is an effort of both Anne-Marit Gravem and me. Because both projects were subject to tasks from Agnar Aamodt in relation to the TLCPC project, the system design was to be essentially the same.

The specifics of the tasks dictated some differentiation in who did what in the system implemenation however. Where Gravems task was to look at the design for collaboration between two reasoning paradgims, CBR and BN, mine was more in the line of using the system to produce explanations.

Initially we developed each our own system, but based on the same tools. Some time into the development phase we realised that what we were doing was exactly the same thing, and decided to combine our efforts rather than working in paralell. At the time, Gravem had more or less the same as me done, a retrieval system with a case-base. In addition she had modeled a Bayesian network and started filling in the conditional probability tables with information from her source. As there was slight variations in our systems some time was spend to get familiar with the new system.

Specifically my contributions to the system was an extension of the Bayesian network model and the associated probability tables. Additional fixes to the case base and supplemental implementation and fixing of code where needed. The biggest part of my work, however, was focused around the explanation module, which is designed to be on top of the core system, i.e., it is designed so that it does not interfere with the methods which was already implemented.

Aside from writing of code, much of the information in the system is a result of our discussions, and based on feedback from the tester it seems the information benefitted from this in terms of quality.

Since our collaboration happened some time into the development, we had no structured approach to who should take responsibility for what. At the test phase this was pretty clear cut however, since our different tasks more or less dictated the purpose. The testing of the system as a whole is Gravems work, i.e., the testing of the quality and accuracy of the predictions of the system. Some of the tables from this will be presented in the result section (6.1). However, as we both had an interest in the system as a whole, the consultation of the results was done by both of us, giving valuable feedback as to how good our effort had been. During this testing/consultation the explanations were tested on the same person.

1.7 Previous Work

In the fall of 2009 the author did a specialisation project on the same subject as this report about. In more ways than one, it can be viewed as a pre-study for this thesis, where the goal was to propose a design for explanations in a system using both CBR and BN as reasoning modules. Some of the work done for that project has thus relevance to this thesis, and a brief overview of the work done and conclusions made are in order.

The project was a literature study and as such some of the knowledge acquired during that work has been used when working on this report. Specifically, the explanation goals which were used as an analysis tool in the project is introduced in Section 3.1 and used in the results for comparison. Some of the related research studied is also given some space in this report, a few of the systems and some of the background information about case-based reasoning systems. In addition, since the group working on the “super”-project of this one, TLCPC, is the same still, an introduction to them and their work is repeated in Section 2.1.

Neither then nor now was the design of the system for the TLCPC group finished, and some assumptions were made about the workings of the system. The result was essentially a three-part system as seen in Figure 1.1. The yellow nodes consists of a set of clinical guidelines (CGL) currently being worked on by the EPCRC group in Europe. Our initial thought was to have them act as a sort of rule-based guiding system for the network and provide feature details to the cases. Furthermore, the system consisted of a CBR and a BN part. The intention behind the design was that the BN should help the CBR module to solve differences between new and retrieved cases by following causal links in the former. This way of resolving the disagreement between the features would not only provide sound data from the probabilistic network, but also help generate explanations. The arrows between the three parts illustrates which modules are connected to which. Between the

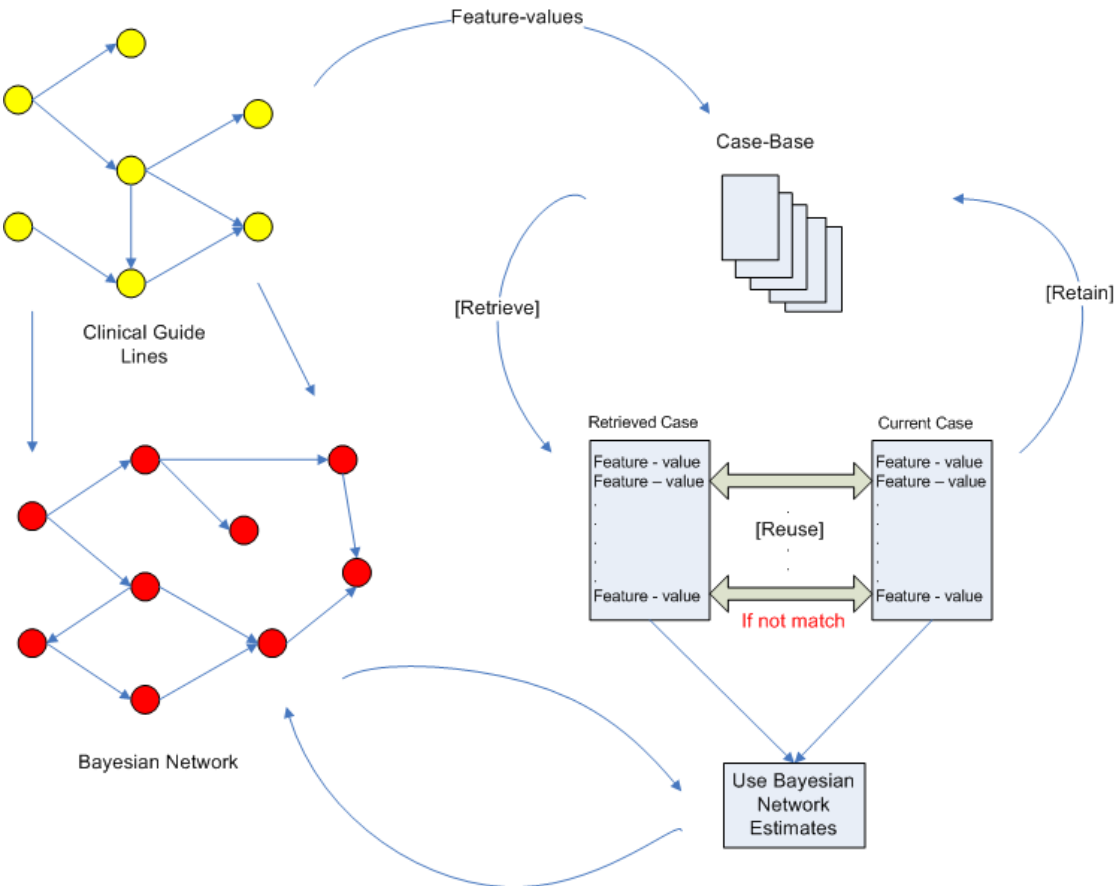


Figure 1.1: System design from specialisation project

BN and the CBR, data flow is shown, where the CBR asks for a solution and get a proposition in return. Between the guidelines and the CBR the arrows depicts information about possible feature-value pairs that are relevant. Internal arrows in the CGL symbolise the decision tree structure of the guidelines, while in the BN they represent causal links.

The result of the study was a design heavily influenced by CASEY. The reason for this was the close semblance between the proposed system and the heart failure program, which is the domain model in CASEY. The most significant difference between the two, at the then current design, was how the data was modeled and represented in the network. For the explanation facility, this particular deviance was not semantically very different, and therefore it could with seemingly very little additional effort be integrated into our system. Data represented as statistical variables would enhance the explanations by containing additional and possibly more accurate data than the semantic links in the heartfailure program.

Some way into the work on this thesis, however, our resident BN expert and bi-supervisor Helge Langseth argued that the approach too closely resembled a semantic network structure, and that it would fail to make the most of the strengths of the BN. The result of this little discussion was a new design proposal, which will be presented in Chapter 4. Because of this decision, my previous design falls on its own premisses, and the results of that previous work will therefore not be focused on. Though, as mentioned, some of the other material will be included.

Chapter 2

Background

This chapter is about the background for the thesis. The first two sections will give reasons for why this particular subject is of interest, introducing the large collaborative project that is TLCPC and the hypothesis. The last part of the chapter will give a brief introduction to the two machine learning paradigms, CBR and BN, which lies at the core of this project, and round off by presenting some systems that are similar to the one presented in this report, or just of particular interest for some property.

2.1 The TLCPC Project

The *Translational Research in Lung Cancer and Palliative Care* (TLCPC) project is an ongoing Norwegian research project with close ties to the larger *European Palliative Care Research Collaborative* (EPCRC) project.

The fact that patients often experience a great deal of pain at some point during the evolution of cancer makes palliative care of good quality important. To complicate matters, when dealing with treatment of pain it has been shown reasonable to believe that each patient experience the pain differently, and at the same time the pain relief with the use of opioid treatment is influenced by genetic polymorphisms in numerous genes (Somogyi et al., 2007), thereby affecting the effect. Due to all the variables it is claimed that a great many of the most prevalent symptoms of the disease is sup-optimally treated today, and thus illustrating the point in creating a computer tool for support of more individually tailored treatment. The idea is that the system will act as a competent “discussion partner” for doctors, giving them access to a wide range of similar cases from which they can get advice.

The project has an emphasis on palliative care of patients in the terminal state of the disease. This involves to a great degree management of the side effects of cancer, and then especially pain, in the best possible way.

The research questions and hypotheses for the project track that handles the decision support system, as given in the project description, are as follows:

- Given the input from the Genotyping and Gene Expression studies, and CAT-based symptom assessment, how to best utilize this knowledge and information in an integrated decision support system for clinical practice?
- A combination of generalized and situation-specific knowledge captures the two main knowledge types in decision-making. How should general models and specific case experiences best be combined to achieve the intended support?
- What is the significance (i.e. usefulness, impact) of such a system in clinical decision making?
- What is the significance of such a system for functional genomic research, i.e. translational genomic research?

The domain of medical diagnosis has been shown to be a complex and uncertain domain where machine learning methods such as Case-Based reasoning (Aamodt and Plaza, 1994) and Bayesian Networks are able to adapt well by utilizing a combination of general knowledge and specific cases. As for this, a computerized decision support tool will be developed with two target-applications:

- Support to the practice of functional genomic research
- Clinical practice of lung cancer diagnosis and treatment

The project of this particular report looks at a sub-component of this system as specified in the project description.

2.2 Hypothesis

The hypothesis of the project overall is that the combination of CBR and BN in these kind of helper, and/or recommender, systems will result in better results. Given the high demand of evidence to support solutions in environments such as health science, a system that can provide suggestions based on both earlier recorded experiences, in the form of cases, and statistical data from the network might offer the robustness needed to make it trustworthy.

In CBR systems the results provided are the conclusion reached by evaluating the stored cases up against the newly presented one and offering the one most alike, compared to some given parameters, as a potential solution to the new problem. In

fields such as health science, where the use of evidence-based solutions are the most common, the reliability of the solution is often in question. In other words, in order for a solution to be accepted as a good one it needs to have the facts right, and in most cases only facts tried and proved right are qualified as “proper” solutions. When this is the case, simply offering a former solved case as a solution, with or without adaptation, might be a questionable approach at best. Having a model based on facts, given in the form of statistical data based on established medicinal methods is a way to alleviate this. To represent this kind of domain information a BN is added to the reasoning system, giving a sound foundation to build the process upon. Even though a case in the case-base might lack the integrity of a proved method, the facts stored in the network will provide the lacking information if needed.

Generally, in a system such as this there is a need for explanations to give the user insight into the reasoning process and to clarify the use or meaning of data. Furthermore, a justification of the solution might provide either a confirmation of the expert’s beliefs or convince the mentioned expert of the validity of the solution provided. Extensive research has been made on explanations in both the CBR and BN paradigms and has resulted in higher user-friendliness over the years since this research started. The hypothesis for the task in this thesis is that by combining the two paradigms, explanations that not only refer to experiences but also to facts stored in the BN will provide a better and more robust explanation to the user. Better because more data has been used and more robust since the statistical data in the network may be verified objectively and not just based on earlier successes.

2.3 Case-Based Reasoning

A case-based reasoner solves problems by using or adapting solutions to old problems

This is a seemingly classic quote from (Riesbeck and Schank, 1989) and is the quintessential of the basic idea behind CBR.

In order to clarify the use of CBR it is appropriate to mention that the expression may refer to two things: The first is the cognitive science perspective on CBR, where studies are made in regards to how humans reason about every-day activities and experiences. The second distinction is CBR-systems, which is a both a research paradigm and a methodology within the field of machine learning (Kolodner, 1993). As a general method for reasoning, CBR bears strong semblance to human reasoning processes where events and experiences are divided into separate occurrences and

stored. When confronted with fresh problems these old experiences are brought forth and compared to the new event, establishing if any of the old solutions might be used to solve the new problem. This particular trait of human reasoning has been empirically studied by scientists from fields both philosophical and scientific, and the advantages of such an approach to machine learning has spawned several CBR systems over the years.

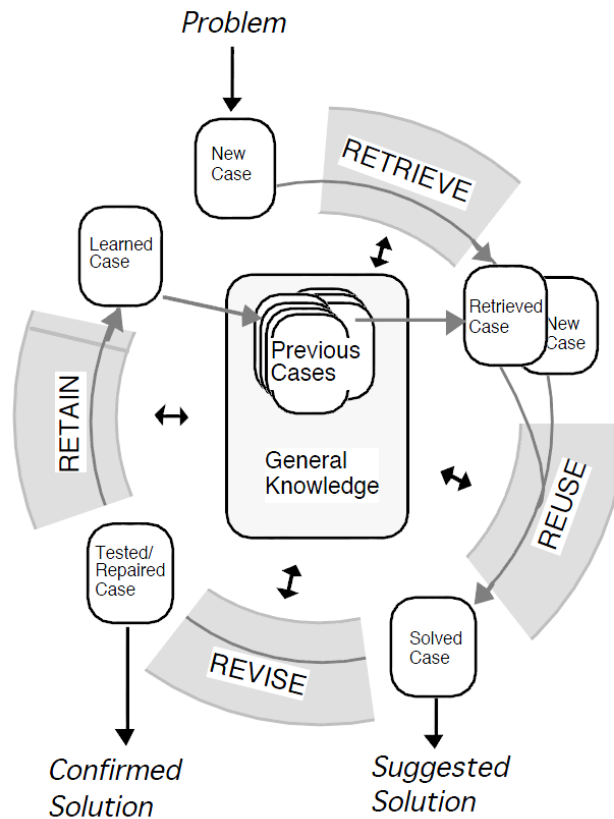


Figure 2.1: The CBR Cycle as presented in (Aamodt and Plaza, 1994)

Figure 2.1 shows the CBR Cycle as presented in (Aamodt and Plaza, 1994). Basically it describes the four general processes that takes place in a CBR system. There are several other and more specific parts to the whole system, of course, but as an illustration for how they operate the cycle holds. There are two main components in the reasoning process, the new problem and the solved problems stored in the knowledge base. Each of these are represented in the form of *cases* which holds all the essential information about a problem. When a new case is presented the system, the first step is triggered. *Retrieve* compares the new case to the previous cases and retrieves one or more. This case is then combined with the initial problem in the *reuse* step, which together will form a solution, e.g. in

the form of the original problem description and a proposed plan of action. *Revise* is responsible for ensuring that the solution works, either by automatic testing or through external guidance. When an acceptable solution has been found the new problem-and-solution case is stored together with the other cases in the *retain* step.

There are basically two distinct types of CBR systems. The knowledge-light systems contain no explicit general domain knowledge, but relies on the information stored in the cases to be enough. In knowledge-intensive systems a model of the world is used together with the case-base in order to solve more complex problems. This knowledge may, e.g., be in the form of rules, semantic network links or, as in our case, a causal network.

One of the main strengths of the CBR approach is that problem solving and learning are merely two sides of the same coin (Aamodt and Plaza, 1994). Each new case that is solved is eventually retained in the system, available for use the next time a similar problem occurs. Not only does this make the system better over time, it can in many situations save a lot of computational time compared to other problem solving methods such as rule-based systems. Each case represents a solved case, and by adaptation potentially a partially solved case. If, say, the new problem turns out to be but a variation of a previously solved case, there is no need to start the reasoning from scratch which saves a lot of time. This is essentially what CASEY (see 2.5) has done. By implementing a CBR part on top of a causal model the system takes the best from both domains.

2.4 Bayesian Network

A Bayesian Network (BN) is essentially a directed acyclic graph (DAG) representing a set of probabilistic variables in the form of nodes. Directed since links connecting the nodes goes from one to another, and acyclic since no loops are allowed, including self-looping.

BNs can be looked at as a systematic way of representing conditional (in)dependency between entities. A defining expression of a BNs features can be said to be this particular property: “A node is conditionally independent of its non-descendants, given its parents”. In many situations when using probabilistic reasoning one is interested in making a representation of the world that best describes all factors. When dealing with real-world modeling, however, a large part of the equation often consists of uncertain variables and the independence property simplifies the process of representing this. Ideally one would like to have a model of the domain as accurate as possible. In health science for example, having a 100% accurate model

of every single disease's progression would make treatment so much easier. The real-world is, however, far from ideal and having perfect knowledge is usually not remotely possible. Therefore it is of great importance to have methods that can produce good results even when operating under uncertainty, and this is exactly what a BN allows us to do.

$$Pr(x) = \prod_{v=1}^n Pr(x_v | x_{pa(v)})$$

The above equation is how the conditional independence property from the quote is represented in more mathematical terms. Simply said it states that the probability of some variable x is equal to the product of all x given their parents. Without this condition one would have to calculate this probability for each variable given all others, which is a lot of work.

In addition to handling uncertainty well, BNs also simplify the computation of large domains by this property. Since all nodes are represented by conditional probability tables (CPTs), each variable might represent a large part of the world without going into too much details. In our case, the reason for why a meal has a strong flavor might be an effect of lots of things; chili, salt, butter, garlic, etc. All of these reasons are taken into account in the CPT, if only just approximately, but it saves a lot of work not having to model every single reason for why it is strongly flavored. In addition, if one is careful when modeling the domain and sustain the cause-effect relationships often found, one can reduce the numbers required in the CPT. If we assume boolean variables the number goes from 2^n to just $n2^k$, where n is the number of variables, and k is the number of variables directly influencing a certain node. In complex domains, this makes the task of filling in the tables go from “nearly impossible” to “almost easy”.

2.5 Related Work

As stated in the introduction, a lot of research has been done on the subject of explanations. This includes work done in psychology, philosophy and artificial intelligence. Within the first of the three the focus is mainly on how humans explain things, why we need explanations, how they are perceived and so on. In philosophy the question is often in the area of; what *is* an explanation. In computer science, and then in particular artificial intelligence, the research revolves around how to make the most of the explanations the systems can provide, and in what manner these should be presented for best effect. Now, these three different takes on explanations are by no means separate, in fact there is a great deal of overlap if one looks at the literature. Much of the theoretical work done in this area touch

upon a lot of different fields of expertise, trying to weave it all together in order to find the best possible way to define, make and present explanations. Following here is a quick run-through of some of the systems that deal with explanations, either within CBR or BN.

CASEY (Koton, 1988) is a CBR system developed for integrated work with a model-based expert system called the Heart Failure Program. The model is a causal network containing the domain information and the proper relations between symptoms and diseases in order to diagnose and patients based on *findings*, i.e. the features and the corresponding values measured. The synergy between the case and the model when evaluating the retrieved case is based upon a causal explanation of the reasoning process and the evidence that supports the final diagnosis. A new case will be feature-to-feature matched with the retrieved case to measure the similarity. When encountering features which are either different from each other or absent in one of the cases, the adaptation step will examine the explanations presented in the reasoning chain. In the first situation, where differences occur, the task is to identify whether the differences are significant enough to rule out the diagnosis or not. The causal model provides an explanation of the findings, and two seemingly unrelated features might simply be e.g. two internal states of the same symptom, and therefore decided to be, in practice, equal. It might also be the case that the particular feature was never used in the reasoning process by the model and is then deemed to be of no importance in the final diagnosis. When presenting the user with the solution, CASEY explains the adaptations that has been done in order to justify the final solution to the problem compared to the retrieved case.

ACCEPTER (Leake, 1991) is a case-based explanation system that was earlier embedded into another such system, SWALE. Both systems are experimental, or research systems, build to test out different approaches to explanation generation based on previous experiences and is thus not used in any practical form. The system is a story-understanding system which characterise anomalies found in stories to facilitate explanation retrieval and adaptation. The focus of this research is how to structure the case-base in order to most efficiently and accurately retrieve and, if necessary, adapt the cases to provide good candidate explanations. As is pointed out in the paper, there will usually exist numerous explanations to a given situation and the appropriateness of the explanation given rely on several factors. Among these are the context, the expectations and knowledge of the “user” and why the situation needs to be explained. The focus of the explanations are why the anomalies that has occurred did happen, and is an interesting approach in order to give valuable explanations.

DIRAS (Armengol et al., 2000) is an application for the diagnosis of long-term

risk factors in diabetes 1 and 2 patients and is meant to support physicians in determining complications for individual patients. The system is based on the CBR method with each patient represented by a separate case. To calculate the risk factors DIRAS uses a CBR method called *lazy induction of descriptions* (LID).

Each case has five distinct description categories; personal-data, basic-diabetes-data, info-patient-consultation, assessment and risk-pattern. Of these two of them are interesting here, the info-patient-consultation and the assessment. The former of these describes the values found in a consultation with a medical professional and the patient and is thus a form of feature-value pairing much used in CBR. The second category, risk-pattern, contains specific pre-described patterns of risks involved in diabetes patients. E.g., a so-called macro-complication may be the specific risk of suffering from a stroke. The application's task is to compare the feature values of the current patient with the ones stored in the case-base and from this classify the current case based on the findings. When presenting the solution to the medical professional the risk-pattern that was estimated using LID contains references to the particular cases in which it found the matching feature-value pair(s). Explicitly there is no explanation knowledge in the system, i.e., a new explanation is not build, but rather the old case is used as one. This category of explanations, the *case as explanation*, is one that is used in many applications of CBR.

As this is designed to be a support system, the type of explanation chosen might be sufficient when used by professionals, however, for the patients this is of little help. To solve this potential shortcoming DIRAS has a report function which gives a summary of the findings in the case as well as clearly stated risk assessment. A solution like this where one splits up the explanation presentation between professional and novice is an interesting approach, and one that could be particularly useful in learning situations. Studies have shown (Mao and Benbasat, 2000) that there is a difference in what each group is interested in. A novice user may need some more information before being able to infer the same solutions as an expert, in this case it would be in order to be able to say something useful about the situation. In addition, an expert user is more likely to find the system useful if it is quick to use (Gregor and Benbasat, 1999), and a sheet of paper with all kinds of information would probably be perceived as less than useful.

David McSherry has developed several recommender systems that has a focus on explanation. *TopCase* (McSherry, 2005) and *ShowMe* (McSherry, 2004b) are two of these. The first can give strategic explanations that tells the user something about the relevance of each attribute, and explain the solution as a comparison between other cases where one or more attributes do not fit. The latter relaxes the constraints posed by the user's query if one or more of the attribute combinations

leads to an unsuccessful retrieval. It guides the user by explaining which attributes are most useful and which constraints may be deleted. In this way the user implicitly gains an understanding and accept the systems reasoning.

INSITE (Suermondt and Cooper, 1992) is a system for explanation generation of the results provided by inference in probabilistic belief networks. INSITE allows the user to examine the reasoning more closely by highlighting the relationship between findings and solution, and by discussing the effects the evidence has on the variables of interest. These effects include information such as which node is affected most strongly by the evidence, why some nodes may not be affected at all, possible contributions and conflicts of nodes etc. Basically this sums up to provide the user with a greater deal of transparency of the system so that the reasoning and conclusions may be perused more closely. This way anomalies in the system compared to the user's expectations can be explained, possibly enhancing both the trust in the system but also the rate of error as a result of blind trust or mistrust.

Many of the systems presented when reviewing BN based systems already in use, shows that a great number of systems are designed to help experts build or troubleshoot networks, like DIAVAL or Elvira, and are not so much end-user oriented explanation wise. The authors of (Lacave and Díez J, 2002) also claim this, that there has been limited research dedicated to how explanations in BNs can help increase the value of the data. However, a few of the systems include some of the properties that are more user oriented and aligned to this system. E.g. INSITE, and BANTER which is based on INSITEs method, PATHFINDER as well as NESTOR are all systems which tries to give the user an increase in understanding about the reasoning and justifying the solution by giving the user access to some parts of the reasoning.

Chapter 3

Approaches to Explanation

The task of providing an explanation to the user is one which can be solved in many different ways depending on a lot of factors. First and foremost the intention of the system dictates what kind of explanations are needed: If the goal is to teach the user something about the domain, justification and an explanation of concepts used may prove valuable, while transparency is of less importance. If the goal is to act as an expert's advisor, transparency of the system is important, and thus an explanation of how the system reached the solution is valuable. Over the years decision support systems and other kinds of reasoning- and recommender systems have become more and more widespread in their use, ranging from diagnostic tools in medicine to holiday planners used over the Internet. Along with this evolution of the field comes an equally diverse set of solutions to how these systems interact with the user, both in terms of content and information returned to help along the way.

Some work has been done in order to collect, organise and generalise various forms of explanation methods and their goal into frameworks, and so as to not re-invent the wheel some of these will be briefly touched upon shortly. The author has not had any success in finding any such frameworks or comparisons for a combination of BN and CBR systems such as the one being done in this project, so it is presented in two parts, first the case-based explanation and then the Bayesian Network explanation principles. Before that, some general explanation goals which are applicable to both types of systems are presented.

3.1 The Explanation Goals

Sørmo et al. (2005) defines five categories of *explanation goals* in order to classify different kinds of approaches to explanation in CBR in particular, but these goals has a broader reach and can equally well be applied to other forms of reasoning systems. As a “benchmark” for the explanation generation in this project, these goals may provide a good pointer as to what is needed in our system. As these are in general valid, an evaluation of the degree of fulfillment of the goals for this system can be found in the result chapter (Chapter 6).

The main thought in the article is that rather than categorising explanation types by what knowledge need be applied as in (Chandrasekaran et al., 1989; Gregor and Benbasat, 1999; Swartout and Smoliar, 1987), the *goals* approach identifies the fact that many of these types might appear in different roles. As is pointed out, an explanation technique might fulfill more than one of these properties and the goal-based approach helps creating a less restrictive categorisation scheme.

The five explanation goals suggested are:

- Explain how the system reached the answer (Transparency)
- Explain why the answer is a good answer (Justification)
- Explain why a question asked is relevant (Relevance)
- Clarify the meaning of concepts (Conceptualization)
- Teach the user about the domain (Learning)

Each of these five have some properties that make them more suitable in some systems than others, and in relation to this project three of them are particularly interesting. The following paragraphs will give a brief overview of the five and argue for why only three will be in focus.

The Transparency goal, *Explain how the system reached the answer*, is important in that it will help the users of the system in understanding how the solution that was proposed was found. An empirical study in Mao and Benbasat (2000) suggests that expert users find this kind of explanation the most helpful in comparison to justification and conceptualizations due to the nature of their own expertise. I.e., because of their knowledge of the domain the experts will quickly be able to reason about the situation on their own, and subsequently form an opinion as to what treatment (in this case) is the most fitting. According to studies made on the subject of explanations, humans require explanations when confronted with situations that don't meet their expectations or outcomes that are anomalous with their own reasoning (ref). The transparency goal fulfills this need for explanation in that it enables the users an understanding of how the system reached the conclusions

it did, and thereby providing a means of identifying where the expectations of the user conflicted with the reasoning of the system. On the background of these findings the transparency goal seems a good fit to the system being considered in this project as the users will be medical professionals and is thus qualified as an “expert” user.

The second goal is the justification goal, *Explain why the answer is a good answer*. This type of explanation will give the users a, simply put, better reason for trusting the system. In other words the system will present the user with some kind of information that supports the conclusion the system reached. The study in Mao and Benbasat (2000) and claims stated in Sørmo et al. (2005) suggests this type to be of most use in systems where the cost of failure is low, and particularly in systems where there is some party involved that “wants” the user to choose a particular case (in this case this might mean movies, books, etc.) like for instance e-commerce providers. And, indeed, providing a justification for why a certain deal is better than some other in this type of interaction might be limited to low-cost domains (the example is that users will not tend to base their decision to buy an expensive vacation package just on such a recommender-system), there is no argument that justification might not be important otherwise. Given the field of medical diagnosis, which is in many cases far from a low-cost domain, justification may provide the users with an extra sense of security when trusting the system if it can give a proper explanation as to why the solution is good. E.g. arguing that in previous cases of some similarity the solution provided was sound, and therefore the proposed solution should be good based on this knowledge might help convince the user. At this point there is a need to point out that the task of the system is not to convince but to help, so the phrasing does not imply an overselling of the solution, but rather the generation of a solid foundation to back the proposition. In fact, given the possible ramifications of the decisions of doctors, a justification type of explanation given might be to present solutions that are similar but of which the results can be shown to have e.g. failed in the past.

The Relevance goal is in this project not viewed as an important requirement of the system. In conversational systems this type will provide the user with an understanding of why the system wish to pursue some path in its reasoning for example, giving an explicit overview of the process itself. It might also be used to give the user the ability to verify the approach used by the system (Sørmo et al., 2005). However, given that the system considered in this project will be used by experts it is the verification of the solution that is needed, and not necessarily the technical details about how relevant each of reasoning parts were.

Clarify the meaning of concepts, or the Conceptualization goal, is in this context something of some importance. Given that the Case-Base will (eventually) be

a repository of knowledge gathered from a large variety of medical experts it is inevitable that there will be terms and concepts used that might have some degree of uncertainty about them for some users. In addition to this aggregation of knowledge, the domain model itself will be constructed by technical experts (although with consultation) and not medical professionals and thereby increasing the risk of using concepts and terminology that not necessarily correspond with what is expected in all cases. The possibility of giving an explanation of this is therefore considered.

The last of the five goals, learning, is not a part of the current “job-description” for the system and will thus not be considered in this project.

3.2 Case-Based Reasoning and Explanation

As stated earlier (see Section 2.3) CBR is intended to imitate the way humans reason in order to solve problems. Since humans also have knack for producing explanations for the solutions, researchers has for some time now looked how to best include facilities for explanation in these systems. Such capabilities in combination with CBR systems has over the years become sort of its own field of research, often called Case-Based Explanation, and there exists numerous systems which has an explanation component. Although many studies have been conducted to show the usefulness of methods, no single best approach has yet been found, and scientists are still discussing pros and cons of each others systems. There has, however, been some aggregation around basic and well-tested approaches to this problem.

In general terms, the work on case-based explanation can be divided by looking at the amount of knowledge-engineering involved in generating the explanations. The *knowledge-intensive* approach produce explanations from patterns or other information encoded in the cases themselves, providing a way for more specialised, dynamic and/or detailed explanations. In *knowledge-light* systems, explanations are normally given by simply highlighting and identifying the key features of a case, or else show the entire content of a case to the user.

Precedent-based explanation is one knowledge-light approach, and suggests that presenting the user with a previously solved case as an explanation can help convince the user that the predicted outcome is a good solution. Empirical study in Doyle et al. (2006) found that users responded well to this type of explanation, and compared to explanations from rule-based systems it gave a more convincing explanation. The key to the approach from that particular test was the use of the *a fortiori* principle. In short it explains and justifies solutions based on a case

that has more severe symptoms but still supports the decision. This particular way of constructing convincing explanations has foundations in human explanation patterns. E.g. if a child wants to ride the roller-coaster in a theme-park but the parents are unsure whether to allow it or not, referring to a friend that is younger/smaller that was allowed to ride is a more convincing argument than referring to one which is older/higher. The *a fortiori* principle has later been used in other knowledge-lite explanation systems (Bridge and Cummins, 2006; Nugent et al., 2007). The approach has some shortcomings, one of them which is commented on in (McSherry, 2004a). Mainly it revolves around the suggestion that if used as a *justification* of the solution, it generally ignores the possibility that some of the features in the retrieved case might oppose the choice, and that because of this the user might be misled by the explanation given. (Nugent et al., 2007) tries to rectify this with the *KLEF* framework, which propose to give an explanation of the nearest *unlike* neighbour of the retrieved case, essentially presenting the user with the evidence which oppose that particular case as a solution.

Interaction-based explanation systems is implemented in a collection of conversational recommender systems where the goal is to provide the user with enough feedback to either accept or decline the proposed solution. Mixed-initiative systems, like TopCase (McSherry, 2005), tries to accomplish this by allowing both the user and the system to ask questions of each other, and by this providing implicit explanations of the choices made by the system. This approach guides the user through an iterative query method, where at each point the system can explain the *relevance* of the questions in strategic terms, and finally justify the solution by showing that changes to attributes will have no effect on the outcome.

The intention of explanations is varied. In CDSSs explanations is a way for e.g. physicians to gain a better understanding of the reasoning in order to be able to more fully trust the system. In e-commerce systems, explanations can be used as a way to persuade users to buy a product. In both cases the goal is to gain **user acceptance** of the system. In their movie recommendation system, *MovieLens*, Herlocker et al. (2000) studied the effect of different forms of explanations aimed at consumers, and the effect on the acceptance of the system. Basically, this type of system is the same as many e-commerce providers use, and the results showed that giving the user a histogram of the ratings of movies by users with similar taste received the best feedback. In addition, a statement of the system's performance in earlier events, e.g. 80%, was perceived as the next most convincing explanation.

The knowledge-intensive approach has also received attention over the years. One of the earliest attempts at defining how to build good explanations from knowledge in the system is found in (Schank and Leake, 1989), where, in relation to the system SWALE an approach consisting of *explanation patterns* (XPs) is presented.

Basically the system generates and learns explanations by looking up a network of beliefs that holds information about various events that may happen. In this case, the particulars of the explanations are stored in the domain structure itself and thus makes the explanation a **part of the reasoning process** itself.

CREEK (Aamodt, 2004) makes use of the general domain knowledge by embedding the cases in the domain model itself, and by this creating a strong coupling between the cases and the general knowledge. The features of the cases are nodes within this semantic network and each represents a concept in the domain model. Each of the reasoning steps in CREEK is designed as an activate-explain-focus cycle, where each step is activated in turn as the reasoning progresses. When solving problems CREEK searches through the knowledge-base and finds the semantic links between features. The links are rated according to their strength, and when presenting the user with a solution these strength-indications are shown as a percentage behind each attribute, essentially giving the relation between the previously solved case and the new one as an explanation. In addition to this calculated similarity the semantic relation between concepts are given as explanation, where one concept may e.g. be an enabler or a subclass of another.

3.3 Explanations in Bayesian Networks

Bayesian Networks (BNs) is a form of probabilistic inference networks that has the characteristics of a directed acyclic graph (DAG), with nodes containing the probabilistic values of the particular property. Each link can, in domains suitable for this, be connected in such a way as to reflect causal properties of the world it models. When humans reason we often view events that follow one another as causal events, where one event is the cause which precedes something that brings about an effect. When using networks to model a reasoning process it is therefore tempting to automatically assign this causal property to the network itself. It is, however, not a property of the BN itself by default.

Bayesian Networks are, as stated earlier, constructed of nodes where each node is connected to at least one other such node through acyclic link(s). “Within” each of these nodes, conditional probability tables (CPTs) are constructed, giving one value for each possible combination of input and output. The value is simply a probability, or belief in some cases, that the given outcome occurs. A classical example is illustrated in Figure 3.1. The example is that of an in-house alarm going off, and the probability of it being a burglary given that Mary or John to calls it in. The parents of *Alarm*, *Burglary* and *Earthquake*, are the cause of the alarm, while *MaryCalls* and *JohnCalls* are the effects of the alarm going off, each

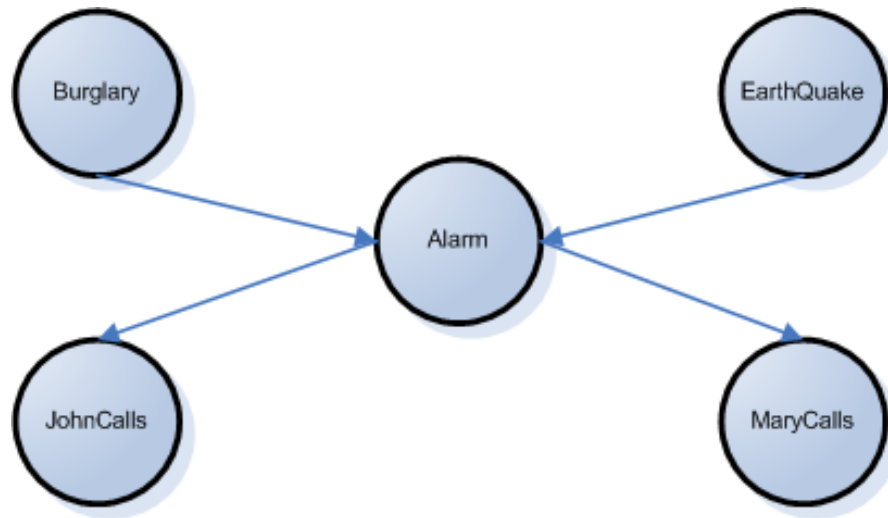


Figure 3.1: An example of a causal Bayesian Network depicting the Alarm network

person having a slight variation in the probability-table as to how often they call in to warn about the alarm ringing when it actually is. The example perfectly illustrates the notion about causal relations and their effect on the design of the BN, and thereby the complexity of the computations needed.

The diagram in Figure 3.2 shows the same network represented without adhering to the causal property of the domain, resulting in links from *MaryCalls* and *JohnCalls* to every possible child. The number of values that needs to be filled in corresponds to a full joint distribution, i.e. 31 distinct probabilities. Not only does this require extra work from the designer- or expert's side, but it also leads to longer computational time.

In the system designed in the context of this thesis there is a strong causal link between the four “stages” of the network, where the evidence is set at the top level, which in turn influence the second and third, which at last gives a solution. This form of cause-and-effect dependencies makes explaining the results generated by the network somewhat easier, as it conforms to the way we humans are used to thinking. E.g. if you were to have steak for dinner, the effect on the solution of this choice would be different from a choice of fish for dinner, which for the wine-enthusiast (and for many of us laymen as well I believe) is common sense and thus relatively easily explained. However, the inverse would not necessarily be true, i.e., if you chose one grape and the solution was a meal it might coincidentally be a grape that worked just as well for fish as for steak.

Over the set of all possible Bayesian Networks the causal property might or might not be a normal occurrence, so as a general rule it is not wise to make the assumption

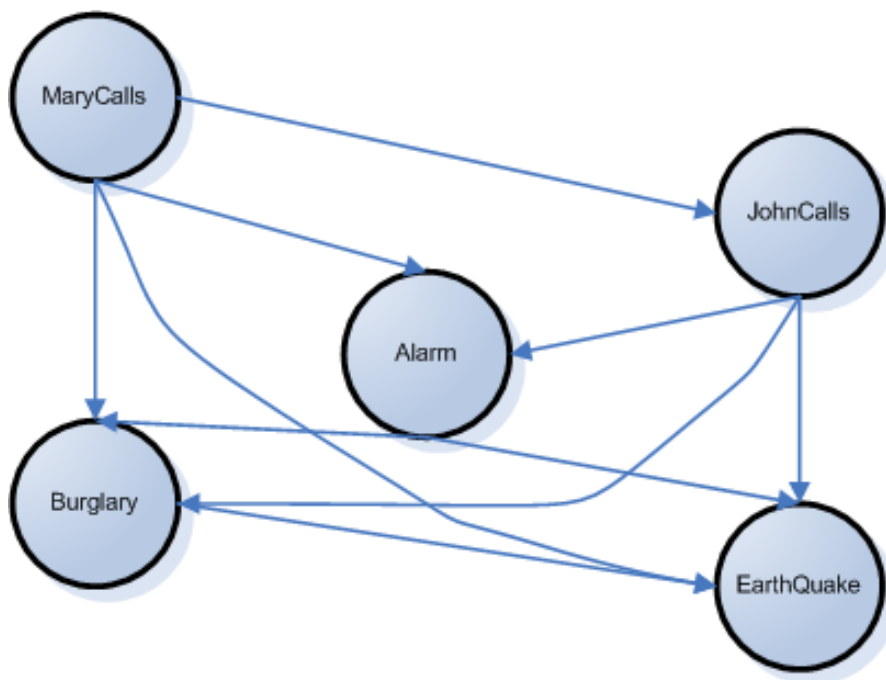


Figure 3.2: A non-causal representation of the Alarm network

of this to hold. However, in the prototype system made for this thesis, as well as in the domain of palliative care, there is reason to believe that causal properties are the norm. I.e., it is possible to follow the reasoning process from the cause, which in health science is e.g. observed symptoms, to the effect, e.g. a disease, but not necessarily the other way. Thus will the explanations provided in the prototype system's setting to some degree be generally applicable to systems in which a causal relationship between nodes exists. As humans tend to favor the cause-effect way of connecting events this leads to the possibility of more intuitive explanations than a non-causal system design would have.

Although the cause-and-effect relation of the network leads to a more intuitive design than one would get from a non-causal network, there is no way past the fact that exact probabilities like those found in statistics, as well as notions of prior/posterior odds and likelihood estimates in the BN, is something foreign and somewhat incomprehensible for persons not well-versed within the field. However, in high risk communities such as health science the exactness of the information is in many instances more important than an easy to grasp explanation, and especially in regards to probabilities. Research has been made on the topic of linguistic versus numerical presentation of probabilistic data, and even though there has been established conventions as to how best convert numerical linguistic to their

verbal “counterpart”, as in (Elsaesser and Henrion, 1990), this might not be in the best interest in all cases. A. Nakao and Axelrod (1983) argue that the difference between what is perceived as the probability in a statement such as *Often* has a much too inaccurate definition between persons to be a reliable measurement in reports. Following this study one could argue that a simple presentation of the numerical data from the nodes leading up to the conclusion might be a better choice rather than a “verbal” interpretation of the same numbers, even though the last method provides an easier to read explanation.

Table 3.1: The ten explanation properties from (Lacave and Díez J, 2002)

Category	Property	Options
Content	Focus Purpose Level Causality	evidence/model/reasoning description/comprehension micro/macro causal/non-causal
Communication	User-system interaction Display of explanations Expression of probability User’s knowledge	menu/predefined questions/- natural language dialog text/graphics/multimedia numeric/linguistic/both no model/scale/dynamic model
Adaptation	User’s knowledge about the reasoning method Level of Detail	no model/scale/dynamic model fixed/threshold/auto

In (Lacave and Díez J, 2002) a framework consisting of ten properties of explanations in general has been used in order to analyse the explanatory capabilities of various systems using BN as a reasoning method. The table presented there is shown in Table 3.1. The three categories separates the context of what, how and to whom the explanation is intended to be used respectively, and gives a broad look at options for what angle to design explanations.

To tie it into context, one could look at this is as more concrete proposals as to how to fulfill the explanation goals presented in (Sørmo et al., 2005) and mentioned in the last section. At the same time, some of these are specific for networks and benefits from a more in-depth discussion and comparison. The following paragraphs will present a brief overview of the properties that has a use in relation to this report.

First off, our system focus on **explaining the reasoning part**, and not so much on the evidence or model. Evidence is set by the user, in the same way that

symptoms are set as evidence by doctors, so there is no need to explain that part, while an explanation of the model is mostly of use for designers and experts when first constructing or testing the network. As an illustration, the complete BN for the system that one day will be used in palliative care will probably contain dozens of nodes and scores of links between them. When using the system, the model is in the background working and an explanation of this will only hinder the users from doing their jobs effectively. For debugging purposes, using a third-party system such as Elvira's explanation capabilities would be a better choice than to include such facilities here.

On the other hand, explanation of the reasoning part is an invaluable tool for the user as it helps verification of the solution for expert users. In evidence based fields such as health science, it has been noted that physicians are reluctant to rely on the advice from a system without understanding how it was obtained (Teach and Shortliffe, 1981).

Since the system aim at proposing solutions that are solidly grounded in both past experiences and statistical data, the purpose of the explanations will be to **describe** the result through the intermediate results, such as the characteristics of the food in our case. It does so at the **macro** level, where the main lines of reasoning are important, not variations within nodes.

The BN is modeled to be causal, as is many systems relying on models from the real-world. There are many good reasons for choosing a causal approach to modeling when possible, and a comprehensive list is given in e.g. (Lacave and Díez J, 2002, sec. 3.1.4). In essence, it both conforms to the way humans interpret events and therefore makes it easier to model the domain, as well as making the best use of the dependency properties inherent in BNs, allowing for easier knowledge acquisition and possibly easier computation.

Bacchus is not designed to give the user any options for dialog with the system, like e.g. MYCIN (Shortliffe et al., 1975) or PROTOS (Bareiss et al., 1988). The explanations are presented verbally as a combination of explanatory text and numbers from both the BN and the CBR similarity between the query and the solution. In order to enhance the explanation, it might be useful to present the user with a graphical representation as well. E.g., an overview of the BN would have given experts good indication for how the reasoning worked. (In fact, the wine-domain expert used when testing *Bacchus* immediately grasped the idea behind the BN by just looking at an illustration such as the one in Figure 4.5.) In a more complex domain however, the usefulness might have been overshadowed by the sheer number of links and nodes, making the presentation overwhelming rather than helpful.

Finally, the explanations presented do not allow for adjustment of the user's prior knowledge of either the domain or the reasoning mechanisms. It is **static** in this sense as opposed to dynamic models that differentiate between e.g. novice and experts. This choice allows the explanations to be tailored to the user group which are experts within their field. However, if the system were to be used for educational purposes, the level of detail and the methods providing the explanations would benefit from being specialised for those purposes.

Table 3.2: The explanation properties for *Bacchus*.

Category	Property	Options
Content	Focus	reasoning
	Purpose	description
	Level	macro
	Causality	causal
Communication	Display of explanations	text
	Expression of probability	numeric
	User's knowledge	no model
Adaptation	User's knowledge about the reasoning method	no model
	Level of Detail	fixed

In Table 3.2 the ten explanation properties from (Lacave and Díez J, 2002) is revised and only the ones used in *Bacchus* is shown. These will be revisited again in the result and discussion sections.

Chapter 4

System Solution

The system designed here is a prototype of how a combination of the CBR and the BN paradigm might work together to form a better reasoning process. It is not the only possibility and not necessarily the best way of combining the two, but further research on that is currently being done by a Tore Bruland. The Chapter starts off by giving an introduction into the choice of domain and a top-down look at the system designed. The last part of the chapter will go into more details about each of the reasoning mechanisms implemented, as well as the explanation process.

4.1 The Domain

The name of the system, *Bacchus*, is the roman name for the god whom was the god of wine and intoxication. The domain of the system is food and wine, more specifically it is the combination of the two.

Originally, and as indicated in the project description, the intention was to use a data set provided by TLCPC and thus related to palliative care as a knowledge-base. However, the data- and case- structures were not yet complete and the inclusion of these into the project was dropped in favor of a set chosen by the author.

The first set used was a collection of wines and their properties previously used in (Cortez et al., 2009). The set is made up of over 6000 instances, where about 1600 is red wine and the 4900 others are white wine, with information about the wine's properties such as pH level, amount of residual sugar, acidic level etc. These objective values are complemented by a quality metric which is provided by experts. In our effort to make these data work for us we realised that it would be very hard to create a model which reflected the properties on the background of us knowing

next to nothing about the subject. As an alternative, Tore Bruland suggested to create a model of the relation between food and wine, which is also the one we ended up with using for *Bacchus*.

The data set which represents the domain consists of a case base with about 60 cases along with a causal network model in the form of a BN. The case information is extracted from Apéritif¹ and is a collection of recipes together with a recommendation of a specific wine provided by an expert. The information in the BN is extracted from Vinmonopolet², which is a state-owned chain of stores that has monopoly on the sale of wine and spirits in Norway.

4.2 Design Overview

The system is designed to combine the two reasoning paradigms of case-based reasoning and probabilistic inference in order to give as accurate a result as possible. The CBR part of the system stores cases in a case-base with saved recipes and their matching wine as content, while the BN has a causal model of the dependencies between a meal and the most fitting grape type.

While working on the specialisation project on this subject last fall, the basic idea was to have the BN work as a type of semantic network where unknown information was looked up and filled in by following causal links. Basically what would happen was that a new case would be presented the system which would do the regular CBR attribute matching based on some given similarity metric. When encountering attributes that was not found in the retrieved case(s) it would send the attribute to the BN which would in turn follow causal links to see if the given input was either a generalisation or a specialisation of the particular attribute. If either was the case, the probability for them having the same cause or giving the same effect would be returned, resulting in a matching value based on this probability. This way of designing the system was one which was followed for some time into this project as well. However, at one point, after a few months, we discovered that this approach would bear too close semblance to a semantic network and thus fail to utilize the potential of a BN, making it a moot point to even implement a BN in the system in the first place as a semantic network would do a better job of fulfilling that particular task.

Figure 4.1 shows an illustration of the two reasoning systems, the CBR and the BN, and their interaction. Since only the first step of the CBR cycle is implemented,

¹<http://aperitif.no/>

²<http://vinmonopolet.no/>

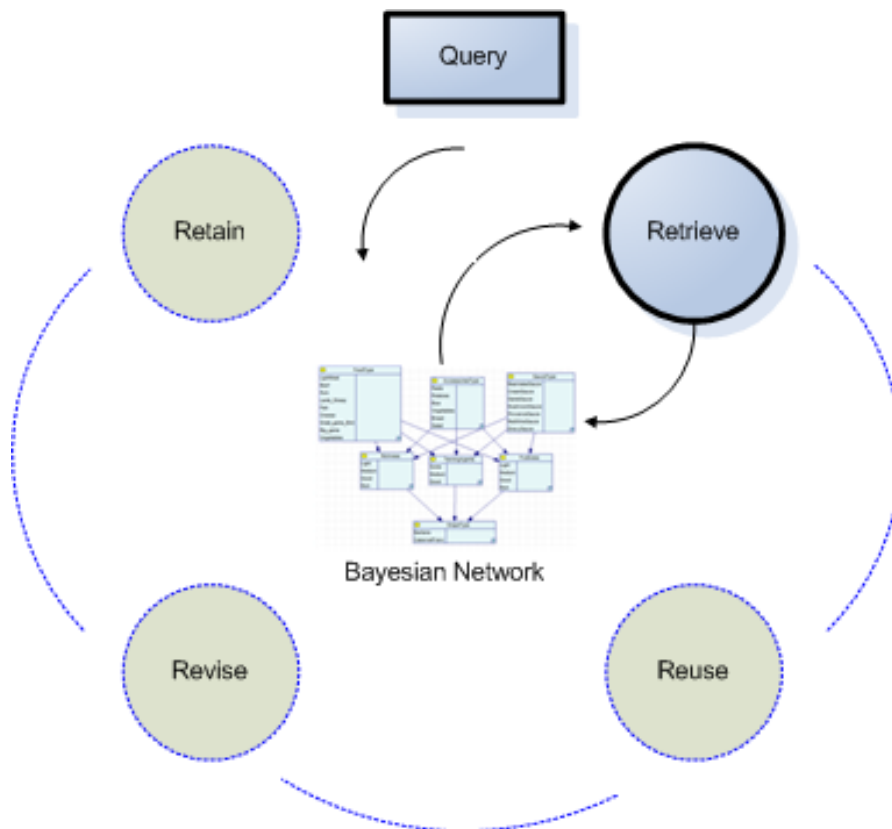


Figure 4.1: An illustration of the system's overall design. The Retrieve step makes use of the Bayesian Network in order to choose the right cases to retrieve.

the retrieve step, the three others are grayed out and connected by stippled lines.

The arrow going from the query to the BN represents the data-flow between the two parties. After the user has created a new request in the form of a meal, the attributes of the features, e.g. that `FoodType = Tenderloin`, are sent to the BN. This is what provides the *evidence* for the reasoning, meaning that these features are set as given, i.e. 100% probable. The evidence nodes in the BN are the top-nodes and have no other parents, only children, and is thus independent of both each other but also the rest of the network. When the reasoning process is done the BN provides the retrieve step of *Bacchus* with a list of each target node's probability. This is what the arrow between the BN and Retrieve signifies. When reasoning is done and the system has a solution prepared, the cases retrieved are sent to the GUI layer and presented the user.

What we ended up with instead was a combination of subjective and objective statistical data on how the various food-, sauce-, and accessories-types influence the properties of the meal (flavor, bitterness, etc.) and in turn what these properties require of a red wine in order to be a good fit. The three properties of red wine; Richness, Fruitiness and Tanning Agents are common description features for a generalised summary of the eleven or so properties of red wine which includes properties such as; volatile acid, residual sugar, alcohol, and pH level.

The common description features are the ones used to describe a wine in the lists provided by *Vinmonopolet*, and which can also be found in the descriptions at their web-site, vinmonopolet.no. This readily available information is what forms the basis for the final result, as the values given the grape nodes in the BN is directly extracted from this catalog of wines. The particular combination of the description features along with a general recommendation of what food each wine goes best along with forms the expert opinion of the network, and thus a recommendation for a type of grape based on this should be a good fit in general.

On the other hand the three attributes that is influenced by the case input; flavor, bitterness and fruitiness, are subjective values, or in better terms, a reflection of our *belief* in how each input value affects the overall composition of the meal's properties. These values will necessarily need tuning over time since the accuracy of the beliefs are currently unknown. However, as they are based on common sense and in large part subject to the individual palate these would be values that can benefit from training the system on new cases and solutions.

4.3 Cases

A case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner.

The quote is from (Kolodner, 1993) and gives a short and precise description of what a case represents. In *Bacchus* the case represents an individual dinner course together with the knowledge of what wine is particularly fitting to that exact meal. The reasoner use these previous experiences in order to achieve the goal of proposing a wine that will fit well with the new case as well. Each of the cases in *Bacchus's* case-base consist of seven feature-value pairs, where three of them are used in the query as well.

Table 4.1: The data structures in *Bacchus*

(a) The case feature disposition.

	Query-Case	Stored Case
Description	FoodType AccessorieType SauceType	FoodType AccessorieType SauceType
Solution		RecipeTitle WineTitle WineOrigin GrapeType

(b) Two example case as they are stored in the database

Case Id	Recipe Title	Food Type	Accessories Type	Sauce Type	Wine Title	Wine Origin	Grape Type
28	Lamb roast	Lamb Roast	Vegetables, Mashed Potatoes	RedWine Sauce	Domaine d'Andérzon 07	France	Grenache, Syrah
48	Grilled clipfish	Clipfish	Fried PotatoSlices	Tomato Sauce	Flor de Crasto 07	Portugal	Tempranillo Syrah

Table 4.1a shows the disposition of the various features. The three features shown in the *description* row; FoodType, AccessoriesType, and SauceType are the ones also used for similarity measuring by the CBR process. The description of the query-case and the stored case are equal both in terms of name and information held since they are feature-to-feature matched.

The four features in the *solution* row are the features used to represent the solution proposed by the *Bacchus*. Obviously these features are only present in the previously solved cases and not in the new one. The *RecipeTitle* is the name of the recipe represented by the case, and is the same title as the one found on *aperitif.no* where the case information has been collected from. The thought behind this is to allow the user to find the exact recipe which is readily available for everyone to use on the Internet. The two next features, *WineTitle* and *WineOrigin* are representations of a specific wine found in the shelves of your local wine store. The last feature listed in Table 4.1 is *GrapeType*. The value, or values as it can be more than one grape type in the wine, is a partial reflection of the solution given by the BN. As noted, the BN provides restrictions for the CBR process as to what kind of grape types the solution should contain according to the beliefs stored in the network, and *at least* one of the grapes in *GrapeType* equals one of the top *i* grapes found as solution in the BN.

<< 35 -> 0.5555555555555555 (1/3) >>

Description	
Food	BeefSirloin
Accessories	[Garlic, Mushroom, Onion]
Sauce	BearnaiseSauce
Solution	
Recipe title	Bacon beef
Wine title	Brown Brothers Cabernet Sauvignon 2003/2004
Wine origin	Australia
Grapes	{CabernetSauvignon}

Figure 4.2: A solution represented in the form of a case with a description- and a solution part.

In Figure 4.2 a screenshot of the solution shown to the user of *Bacchus* is given. As one can see, the entire contents of the solution case is given, representing a specific recipe together with the recommended wine. From an explanation point of view, this is categorised as a **precedent-based explanation** (see Section 3.2), where the solution case is presented in full detail. One notable difference from the features listed in 4.1 is the presence of the similarity value at the top of the screen,

between two arrows that lets the user navigate between the k most similar cases retrieved.

4.4 The Case-base

The case-base of the system is constructed by executing a simple SQL-script at system start-up, more specifically in the `configure` method called from the main method in the `MealRecommender` class. The SQL code is located in a text-file and read, starting the construction of a database through the Hibernate middleware. As a database manager system Apache Derby is used on localhost for this prototype's purposes.

The case-base, shown in Figure 4.3, is a relatively simple relational database with links between `mealId` and `grapeId`, as well as between `mealId` and `accessoriesId`. The former is there for obvious reasons, to be able to link a specified wine to a meal, while the latter is there in order to allow the user to choose between many different accessories from an ontology in itself and in order to enable one meal to contain more than one accessory.

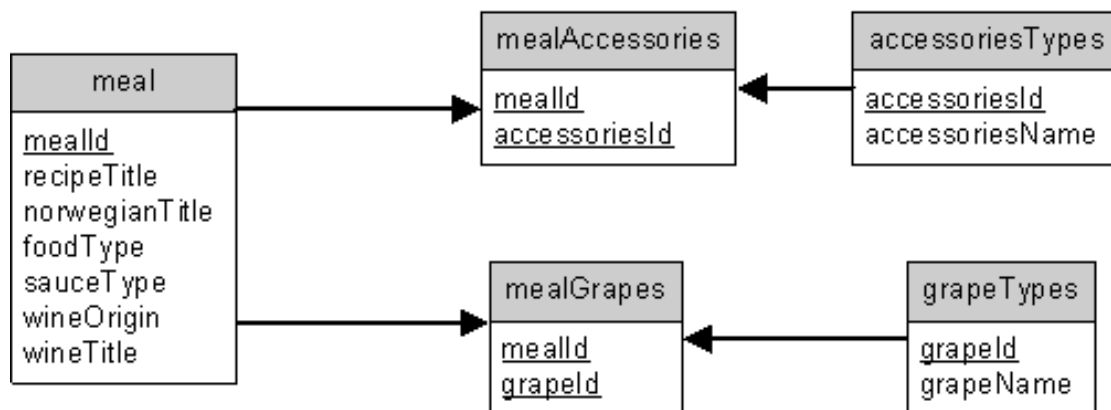


Figure 4.3: The database tables and their relations (Source: Anne-Marit Graven)

4.5 The Case-Based Reasoning Part

The CBR part of the system is made using the classes and methods available in the jColibri framework as described. Within the main class, `MealRecommender`, a cycle method manages the main part of the reasoning process outlined by the 4

re's described in Section 2.3. Of the four, only the first one, retrieval, has been implemented in this prototype system. While the last three stages of the CBR cycle are just as important to the process the task of implementing a full reasoning system is beyond the scope of this thesis.

The retrieval step takes as input from the user three features. `FoodType` and `AccessorieType` are chosen from an ontology tree-view, while `SauceType` is chosen from a drop down menu in the first screen. In this system, these three features comprises a full meal and the combination of the three is used in order to find a suitable match among the solutions stored in the case-base. The core of the retrieval step is the similarity metric used and in our system these metrics are chosen to complement the type of knowledge stored in the domain-ontology. Further details are given in the next section.

The Similarity Metric

jColibri, and thus our system, operates with two definitions of similarity metrics. One is the *global* scoring function which is applied to description- and compound objects, and in *Bacchus* this particular metric is set as the **Average** function. The global metric is applied to all of the three features after their relative similarity has been calculated, effectively this means that the total similarity score is divided by three. It would have been possible to “group” features together in compound objects and have separate global metric for each group if there was a need for it.

The second type is the *local* scoring function, which is the one the user choose in the second screen, and defines how the feature-to-feature similarity between the query and the retrieved case should be calculated. *Bacchus* allows the user to choose between several scoring methods for each feature. The first of the available metrics is **Equal** which is basically a string comparison between the features from the input and the retrieved case. When working with an ontology this is not particularly useful as the result give no real information about the similarity of data represented by the features. Since the sauce is not part of the ontology, however, a custom similarity table has been constructed which reflects our beliefs in how equal each sauce is. The equality metric might also be used, which will result in either a 1 if the same sauce is chosen in both cases, or a 0 if they're a mismatch.

The two distinct choices for the ontology based features, *FoodType* and *AccessorieType*, are `OntDeepBasic` and `OntDetail`. In jColibri these are called concept based similarity functions, and have the following formulas for computation:

$$f_{DeepBasic}(i_1, i_2) = \frac{\max(\text{prof}(LCS(i_1, i_2)))}{\max_{C_i \in CN}(\text{prof}(C_i))} \quad (4.1)$$

$$\begin{aligned} \text{detail}(t(i_1), t(i_2)) = \\ 1 - \frac{1}{2 \times ((\cup_{d_i \in t(i_2)}(\text{super}(d_i, CN))) \cap (\cup_{d_i \in t(i_1)}(\text{super}(d_i, CN))))} \end{aligned} \quad (4.2)$$

Where:

CN is the set of all the concepts in the current knowledge base
 $\text{super}(c, C)$ is the subset of concepts in C which are superconcepts of c
 $LCS(i_1, i_2)$ is the set of the least common subsumer concepts of the two given individuals
 $\text{prof}(c)$ is the depth of concept c
 $t(i)$ is the set of concepts the individual i is instance of

Basically, these similarity metrics calculates the distance in the ontology tree between the query feature and retrieved case's matching feature. Detail has a higher tolerance for concepts that are far apart than fDeepBasic, making it less coarse in the evaluation. However, in *Bacchus* the ontology is in most cases not deep enough to give any real difference between the two.

4.6 The Bayesian Network Part

The Bayesian network in our system is responsible for selecting which grape that suits the current query the best by reasoning about probabilities in the network.

The first design of the BN is shown in Figure 4.4. The main difference between this one and the final design is the lack of nodes describing the properties of the food. The reason for the inclusion of these is basically that there was no way to properly model the influence of Accessories and Sauce on the result. The probabilities given in the nodes for wine properties are extracted from vinmonopolet.no, and is thus tailored to the relation between food types (game, ox, chicken etc.). Simply said it was no good way to estimate the probabilities of the two other evidence nodes without affecting the “expert” values. Another problem found with the original model was that there was very little discrimination between the target nodes, i.e. every grape had probabilities ranging from 5% to 12%. When adding the intermediary nodes the hope was that these would lead to a stronger bias toward just some of the grapes. After running some tests on the new design we discovered that we didn't get the desired effect, and that other measures would be needed for this.

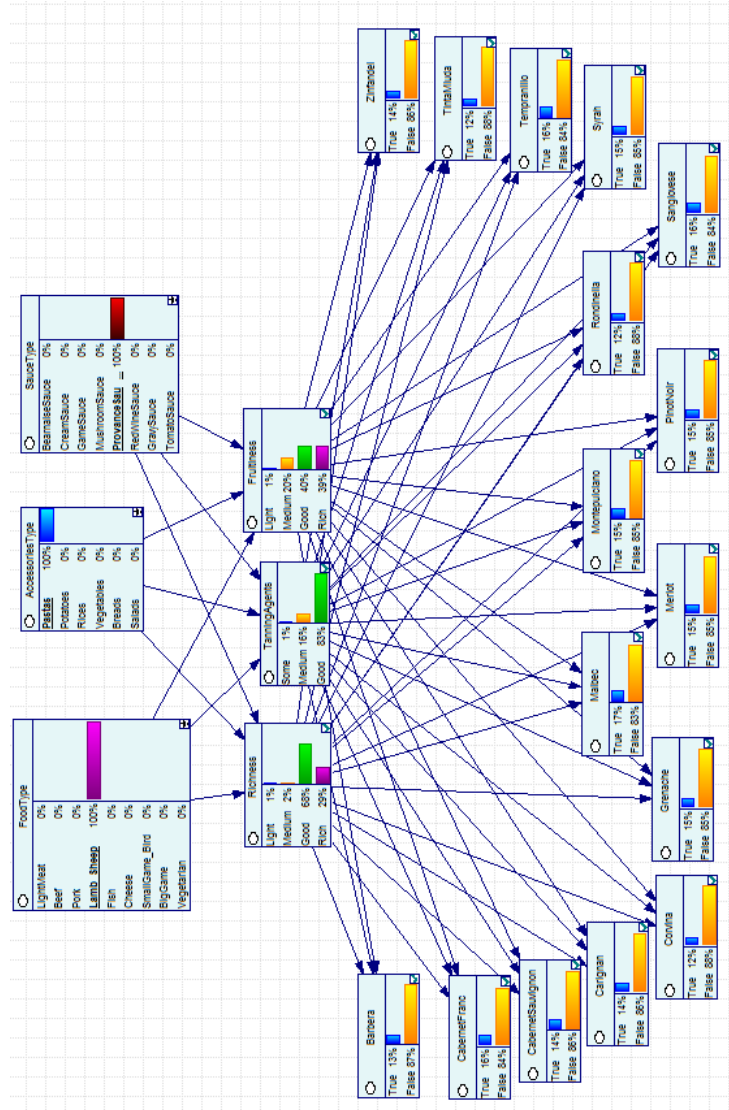


Figure 4.4: The original design of the Bayesian Network. The three nodes for describing food properties were not yet added

The extension property nodes are implemented as Bitterness, Flavor and Freshness, and describes general attributes of food. Bitterness is the bitter taste given by something acidic, like lemon or tomatoes. A meal with high bitterness need a wine with higher levels of tanning agents than one with less bitterness in it.

Freshness is the meal's "weight". E.g., a meal with a heavy cream sauce and lots of red meat is heavier, and may seem less fresh in taste, than e.g. a pasta salad. Because of this, a meal with high freshness can come away with a lighter wine, while the steak would need a wine with more fruitiness and richness.

Flavor is pretty self-explanatory, in that it describes the strength of the flavors in the meal. A heavily spiced meal need a spiced wine with good richness in order to not taste like lemonade. Conversely, a meal which is kind of bland, like chicken fillets with no sauce, need a lighter and more juicy wine.

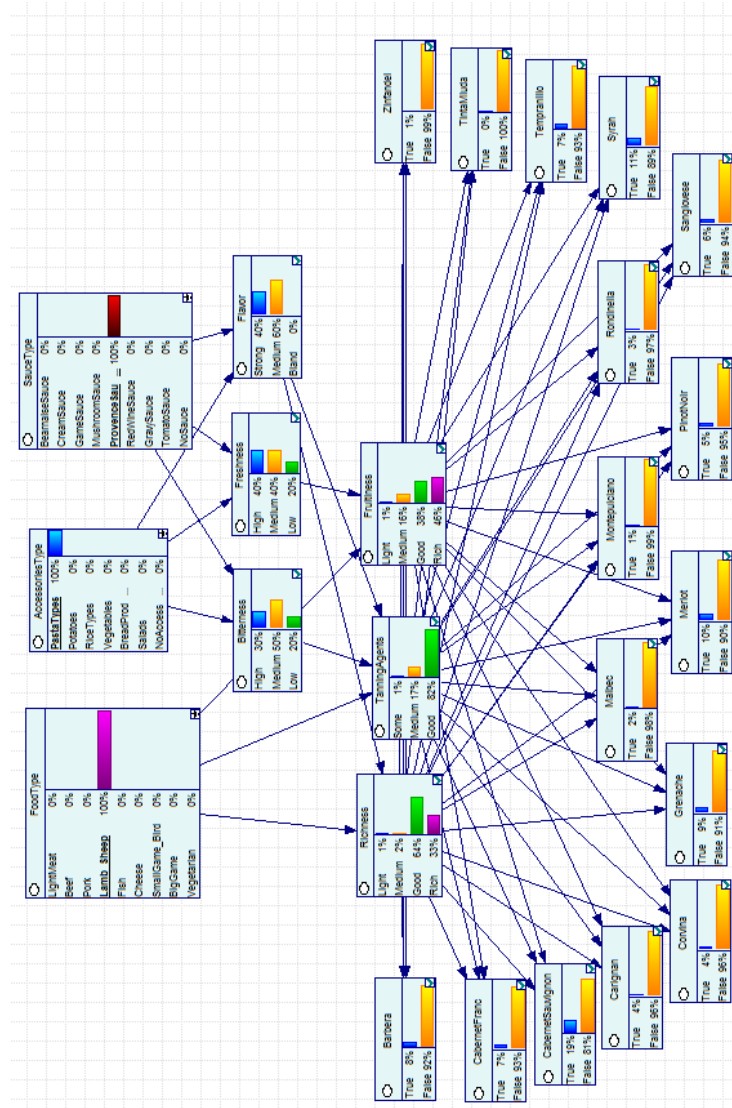


Figure 4.5: The final design of the implemented Bayesian Network

In Figure 4.5 the implemented BN is shown. It consists of 24 nodes, where the three at the top are the evidence provided by the query, the 6 in the middle constitutes the reasoning nodes and beliefs and 15 of them are result nodes in the form of grapes.

Bacchus consists of three layers, where the top layer represent the GUI layer, the middle one is the CBR and the BN system and the bottom one is the database holding the stored cases. In the GUI layer the user is presented with three screens following one another, the first is the query selection screen, giving the user the ability to choose the contents of the meal. The three choices are food-, accessories- and sauce-type where the two former choices are picked from a tree list representing the ontology, and the latter is a drop down menu with pre-stored sauces. These three parts forms the *evidence* used for further reasoning in both the CBR and the BN parts.

When the evidence is set the network propagates the values selected through the network and the result is that the probability of a grape being the right one increases or decreases accordingly. From these nodes the grape-values are calculated in two ways. One is the value given by the network node as is, i.e. the probabilities extracted from the wine overview at vinmonopolet, the second one is a value consisting of the log-odds ratio derived from the amount of increase or decrease the particular grape experienced once the evidence had been set. These two calculations tells us a bit different things. The first one gives the probability that if one was to go into a vinmonopolet store where they had all the listed wines available, the chance that you would pick a wine based on a particular grape that fit well with the given meal (the query) is accordingly to the result. The second method describes what kind of grapes the has an increasingly good or decreasingly bad probability of being the right sort of grape for the particular meal.

4.7 The Graphical User Interface

Much of the code for the Graphical User Interface (GUI) used in the application is reused from an example application delivered with jColibri since it is kind of outside of both the scope and interest in this project. It has been tweaked in order to fit with our application though, but that's also the extent of it since as (almost) everyone who has done GUI development in the Java framework can testify to, it is a lot of work to get it done right.

What is interesting with the GUI however, is how it relates to the rest of the system. When running the application the user has some choices as to how define

the query and the metrics, and these are shown through a dialog. In essence, the user is controlling the progress of the system in that the information provided by each dialog is a vital part of the overall run of the system.

The first of the dialogs presents the user with the query options. The two first, FoodType and AccessoryType are a part of the ontology and is picked through a pop-up window that shows a tree structure of the ontology as it has been defined. The tree structure shown also corresponds to the structure used when calculating the similarity based on the ontology and, depending on the choice of metric chosen, the distance between the nodes act as a measurement. The last choice is the SauceType and is picked in a drop-down menu.

The second dialog presents the user with the choice of which similarity metrics and the corresponding weights to use for each of the three query attributes. This relates, for obvious reasons, only to the CBR part. The specifics of the calculation methods are given in Section 4.5, and will not be further discussed here. What is worth mentioning here is that each of these dialogs halts the system since the query and configuration of the similarity metric parameters is being set by the user. So nothing is happening “behind the scenes” in the system, the user has control. This way of presenting the user with options as new input is needed is something which can be tied into the explanation part of the system. In other words, instead of telling the user to fill in some fields and then presenting the result straight away, the system is letting the user become aware of what is needed in order to complete the reasoning process, and thus one might get a better understanding of the process itself. When the user is done and clicks the next button, the system performs the calculations needed and presents a solution.

The solution presented in the third screen is a textual representation of the cases retrieved as possible solutions. The fields presented the user are a short description of the contents of the recipe, the name of the recipe as well as a specific wine given by its title and year, as well as the grape(s) used. An illustration of the case representation is shown in Figure 4.2. At the top of the screen the user is presented some back and forth buttons used to navigate through the k retrieved cases found to be most similar. Between them the similarity value of the case is shown. At the bottom of the solution screen is a button named “Explanations” which takes the user to a screen providing an explanation of the case currently being shown.

At the explanation screen, three types of explanations are available. The first one is an explanation based purely on the CBR part of the reasoning system. It gives textual feedback on the specific similarity values calculated between each of the features from the query and the solution case. The second one available is an explanation based on the BN. It presents the user with the details of the

probabilities in each node, with a short description accompanying the three “layers” of the network. The first part describes the probable characteristics of the meal in terms of flavor, bitterness and freshness. The second part is the requirements of the wine in relation to the meal’s properties, and the last part is what grapes is most likely to fulfill these requirements.

The last option is to view an explanation based on a combination of both the CBR and the BN. In essence, this part just combines the two and shows them in the same view. This way the user get a better look at what influenced the solution.

4.8 Running Bacchus

This section will guide the reader through the use of the system. The implemented system is attached to this report by a CD. The system can be run through the jar file `bacchus.jar`. The source code is also attached in the folder `Source Code`.

We created a GUI for the system in order to create a more user friendly application. We chose to use the same GUI as `jColibri’s TravelRecommender` in order to reduce the amount of work. This GUI was therefore adapted to suit our system.

Choosing a Meal

The first dialog is where the user selects the contents of the meal. This dialog is illustrated in figure 4.6. `Food Type` and `Accessories Type` are both connected to the ontologies (explained in section 5.2). As we can see from the figure, the Ontology dialog pops up when the user pushes the `Accessories Type` button (or the `Food Type` button). This dialog shows the concepts and the instances in the ontology. Only instances marked with a purple diamond can be chosen. `Sauce Type` is a combo box and contains the same sauces as defined in the Bayesian network (see figure 4.5 in Section 4.6).

After the meal items are chosen, the user presses the `Set Query` button to go to the next dialog.

Defining the Similarity

The similarity dialog (see figure 4.7) is used to defined the similarity. Each of the attributes from the *Query Dialog* has an associated function and a weight.

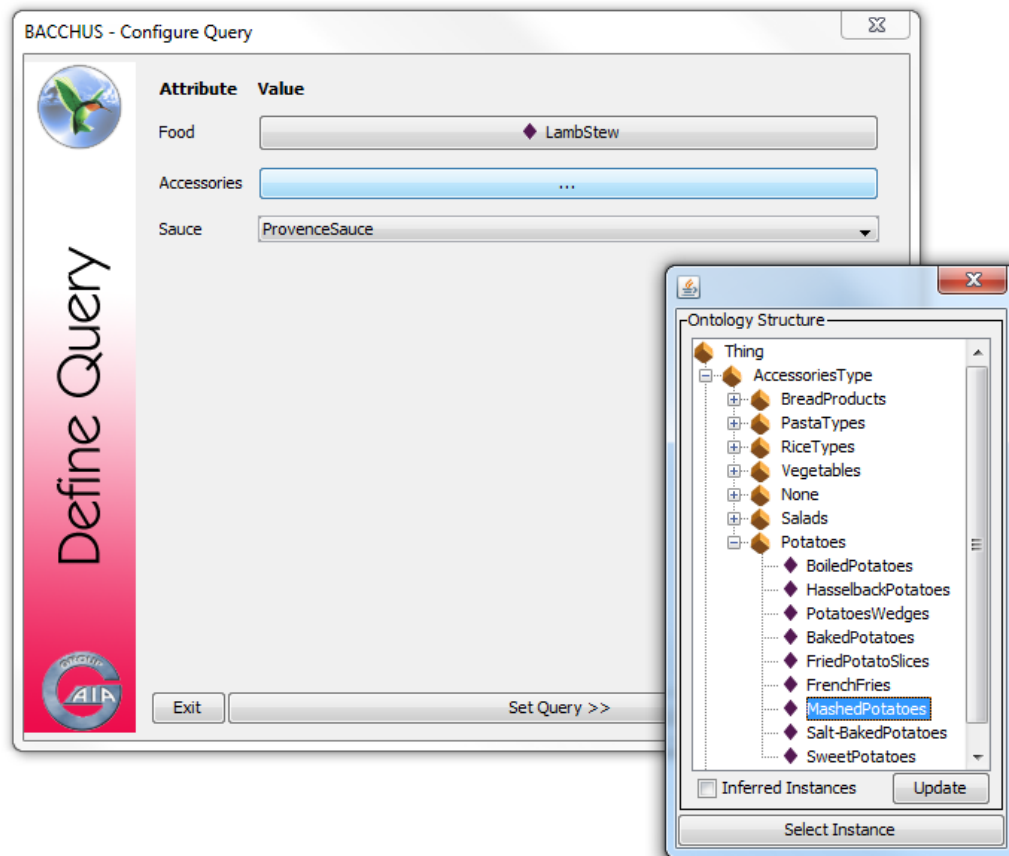


Figure 4.6: Query Dialog

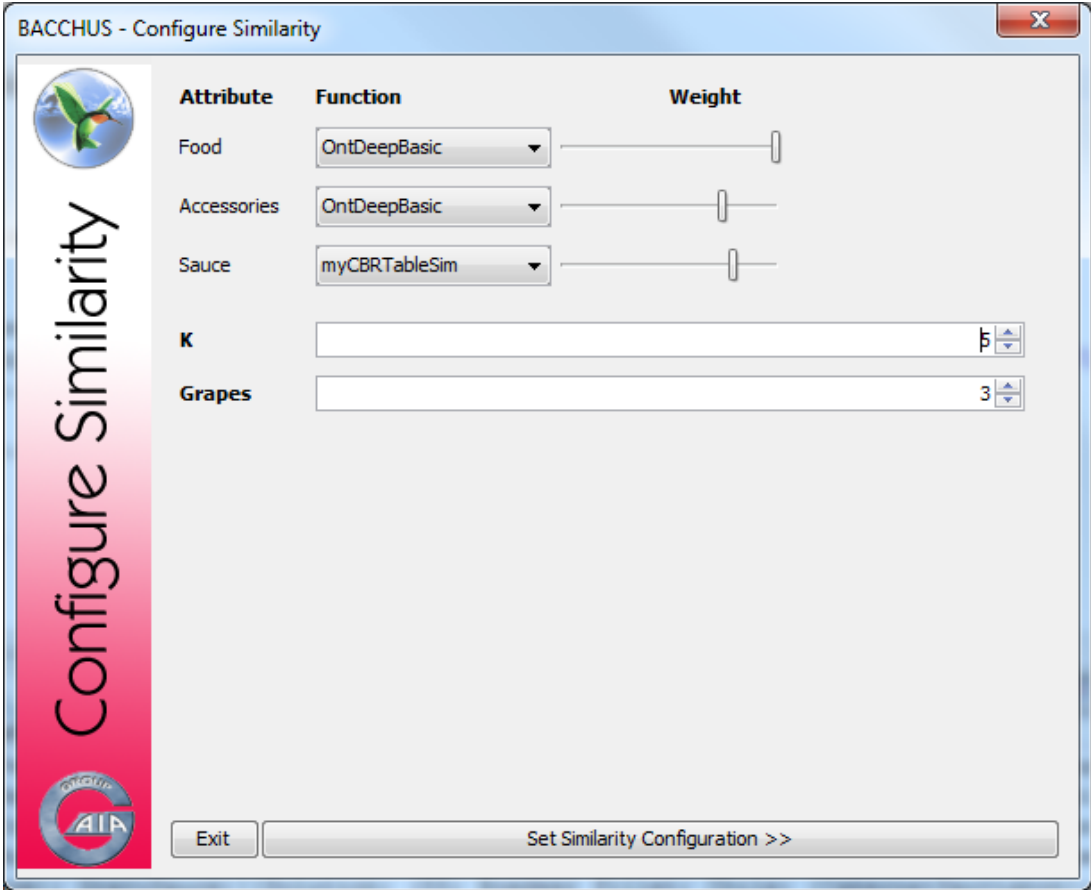


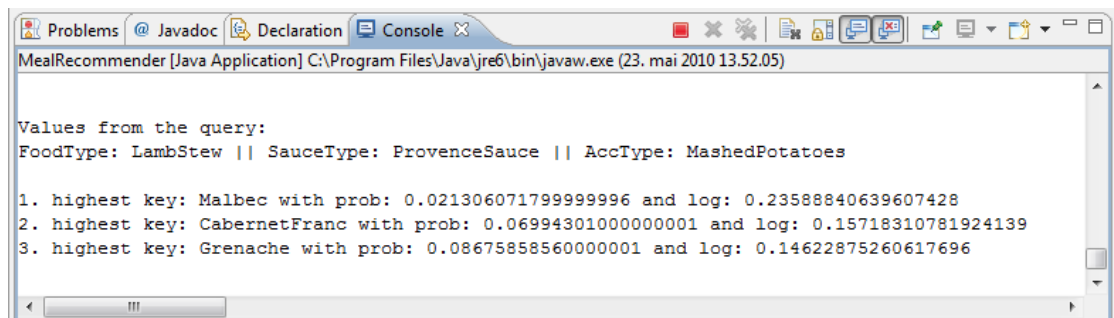
Figure 4.7: Similarity Dialog

As mentioned in 4.5, we have five different similarity functions. **Food Type** and **Accessories Type** are ontologies, and they have therefore special ontology functions (i.e `OntDeepBasic`, `OntDeep`, `OntCosine` and `OntDetail`). In addition, they have associated a function that compares objects: `Equal`. **Sauce Type** is not an ontology, so it has no ontology functions associated with it. Instead, it has the same `Equal` function, and also a special myCBR similarity function: `myCBRTableSim`. As mentioned, each of the similarity functions also has a weight. These weights are used to set the importance of the different attributes in the calculation of the case similarity. It is natural to dedicate the highest weight to the **food type**, since it has the greatest influence on the choice of wine. **Accessories** and **sauce type** should then have lower weights. We also define the k number of cases that should be retrieved and the i number of grapes that should be selected by used in the further reasoning.

The button `Set Similarity Configuration` gets the user to the next dialog.

The Results

Illustration 4.8 shows the console output after the i number of grapes where set in the similarity dialog. The console prints out the most probable grapes and their probabilities. This review of the system is created after the changes done after the initial testing (described in Section 6.1), and the grapes are therefore chosen on basis of the log odds ratio function.



```
MealRecommender [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (23. mai 2010 13.52.05)

Values from the query:
FoodType: LambStew || SauceType: ProvenceSauce || AccType: MashedPotatoes

1. highest key: Malbec with prob: 0.021306071799999996 and log: 0.23588840639607428
2. highest key: CabernetFranc with prob: 0.069943010000000001 and log: 0.15718310781924139
3. highest key: Grenache with prob: 0.086758585600000001 and log: 0.14622875260617696
```

Figure 4.8: The i best grapes

After finding the most probable grapes, the retrieved cases are shown. This dialog is illustrated in figure 4.9. Between the two buttons in the top are some information about the case and the similarity. The first number is the case number, and after the arrow is the case similarity. The numbers in parentheses represents the position of this case among the other retrieved cases.

A retrieved case consists of a description and a solution. It is also possible to get an explanation of why the case was chosen by pushing the button **Explanation** in the lower middle. This is the last step we implemented, so the **next** button will just take the user to a confirmation dialog that informs that the cycle is finished and asks if the user want to query again.

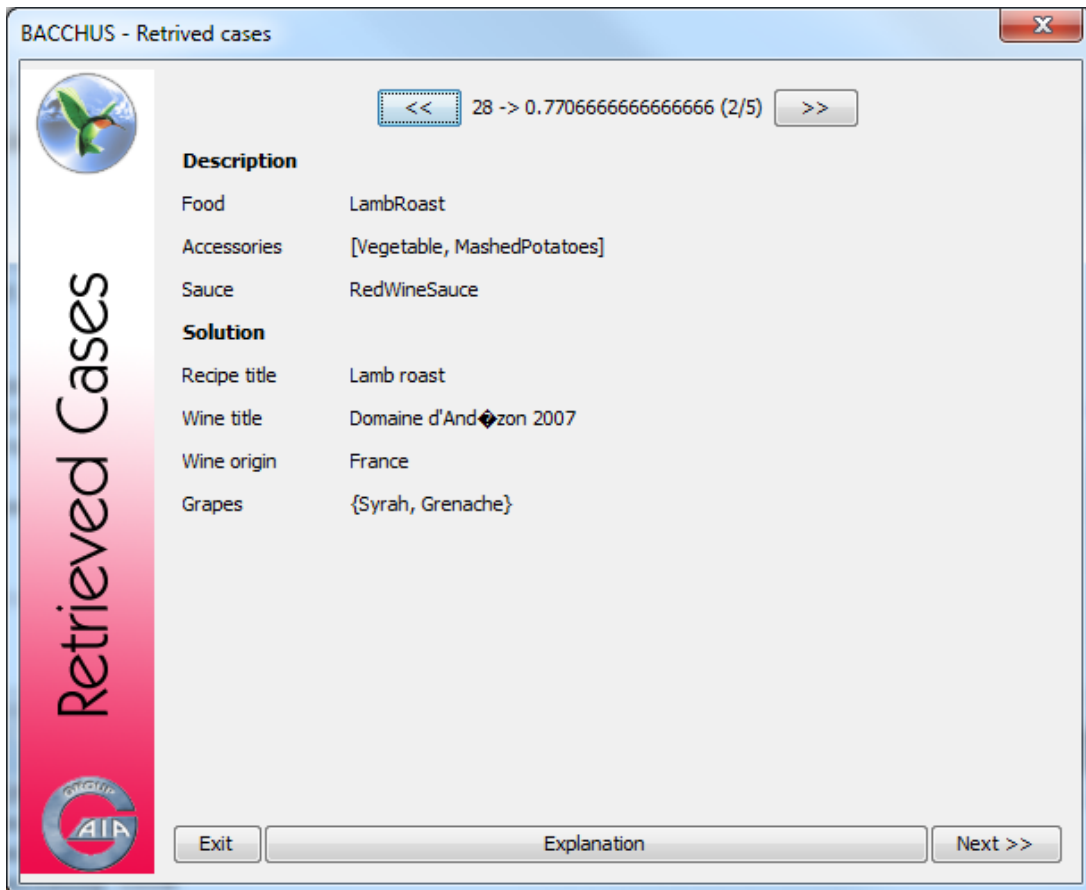
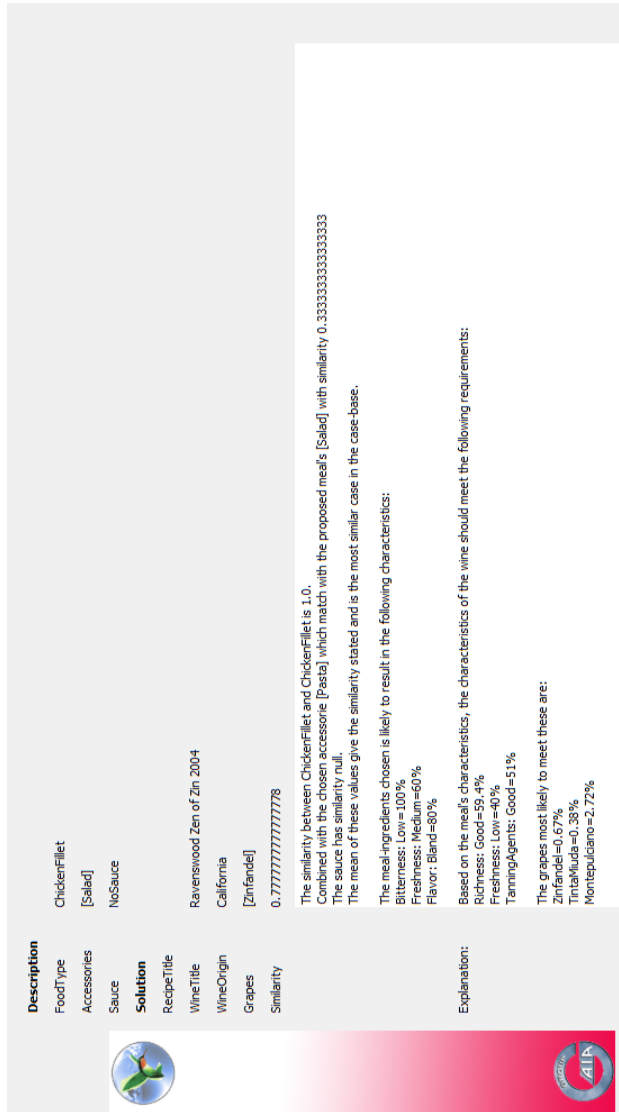


Figure 4.9: Result Dialog

The Explanation

The explanation dialog shows up in a separate window from the result when the appropriate button is clicked in the result dialog.



Description

FoodType: ChickenFilet
 Accessories: [Salad]
 Sauce: NoSauce

Solution

RecipeTitle: Ravenswood Zen of Zin 2004
 WineTitle: California
 WineOrigin: [Zinfandel]
 Grapes: [Zinfandel]
 Similarity: 0.7777777777777778

The similarity between ChickenFilet and ChickenFilet is 1.0.
 Combined with the chosen accessories [Salad] which match with the proposed meal's [Salad] with similarity 0.33333333333333333333
 The sauce has similarity null.
 The mean of these values give the similarity stated and is the most similar case in the case-base.

The meal-ingredients chosen is likely to result in the following characteristics:
 Bitterness: Low=100%
 Freshness: Medium=60%
 Flavor: Bland=80%

Based on the meal's characteristics, the characteristics of the wine should meet the following requirements:
 Richness: Good=59.4%
 Freshness: Low=40%
 TanningAgents: Good=51%

The grapes most likely to meet these are:
 Zinfandel=0.677%
 TintaMuda=0.38%
 Montepulciano=2.72%

Explanation:

Figure 4.10: The explanation dialog

There are three different explanations available from the three buttons shown in Figure 4.10, **CBR**, **Combination**, and **BN**. When either is clicked, the explanation available is shown in the explanation panel.

The **CBR** button gives an overview of the similarity values of the query case compared to the solution. **BN** gives three separate value types back; the first one is the probabilities from the intermediary food tables, i.e., **Freshness**, **Bitterness**, and **Flavor**. The second group consists of the probabilities from the wine characteristics, while the last is the log-odds-ratio calculated for each of the **k** top grapes.

The middle button, combination, simply combines these two into the same window.

The button at the bottom simply close the explanation window and returns the user the result dialog. If the user wish to peruse the other available solution suggestions, he may do so and use the same approach to get explanations for these in turn.

Chapter 5

Implementation

This chapter will give an overview of the architecture of the prototype system that has been build in this project. The first section presents the technology that has been used in development of the system, followed by an illustration and description of how these work together in the technology-stack of the system. The main part of the chapter provides the details about the actual implementation, what kind of methods and functions are implemented and how these work.

5.1 Technology

Within the field of computer-science in general there is usually countless of open-source or commercially available software packages and applications that help designers and developers in doing their job better and more efficient. Within artificial intelligence and machine learning, there really isn't such an abundance, and most systems developed in research is made from scratch. Over the last decade or so, however, there has been progression here as well, and there is now software available to help build reasoning systems too. In the development of *Bacchus* some of these applications has been used to good effect, and these are as follows.

5.1.1 jColibri

jColibri¹ is a Java framework for developing CBR systems that allows users to reuse software code (Bello-Tomás et al., 2004) by providing a substantial library to both make the development more efficient, but also ensure consistency over time.

¹<http://gaia.fdi.ucm.es/projects/jcolibri/>

In our development of a prototype CBR application we used the jColibri framework as a starting platform and extended it with classes to suit our needs. Most prominent is the inclusion of classes to handle the synergy between the CBR and BN parts of the system, as the latter is for obvious reasons not a part of the original framework.

What jColibri allows us to do is to quickly get into tailoring the system to our needs, rather than having to design and implement all of the basic functionality from scratch: E.g. one only has to provide a description of the case structure and how it ties into the case base, as well as constructing a Java Bean for the case class to get a working case retrieval system. Providing this sort of baseline functionality saves the developers a lot of time.

However, the flip side of the coin when using such a framework is that a lot of the classes, methods and functions are buried deep in the hierarchy and thus makes debugging the system a lot harder than it would have been had it all been written by the designer. The obscurity of the exceptions, and especially in a narrow reaching system like jColibri (user wise) makes it hard to pinpoint the exact location and/or nature of the problems encountered.

5.1.2 GeNIe / SMILE

GeNIe² (Graphical Network Interface) is an application for designing networks such as BNs. It provides the user with a graphical user interface that allows for easy and intuitive construction of networks. The user interface is the top layer for the engine, SMILE (Structural Modeling, Inference and Learning Engine), which handles the inference in the background. SMILE provides wrapper class libraries for use with both .NET and Java, simplifying integration with other applications greatly. The Java wrapper, jSmile, provides methods for interaction between the network modeled in GeNIe and the application written in Java.

5.1.3 Eclipse

Eclipse is an Integrated Development Environment (IDE) for Java, an object oriented programming language. Eclipse has over the years become almost the de facto IDE to use for Java development, and offers a stable environment with many handy features to simplify the development process.

²<http://genie.sis.pitt.edu/>

5.1.4 Apache Derby

Apache Derby provides, together with an Eclipse plug in, a simple to use database embedded right into the working environment. Running on a localhost the configuring and data insertion is managed through Hibernate.

5.1.5 Hibernate

Hibernate is an Object Relational Mapping (ORM) middleware that handles the mappings between objects in a programming language such as Java and the underlying storage, a simple database in our case.

5.1.6 Protégé / myCBR

In order to organise the cases and ontology, Protégé with the myCBR plug in has been used. This program allows the system designer to maintain a good overview of the cases, as well as defining custom similarity metrics, like tables, for particular ontologies. In our case, the similarity table for the sauce types has been defined this way since they're not part of the ontology, and the alternative, the String equality measure, does not provide a fitting metric to use in this case.

5.2 Implementational details

The technologies described in the last section are implemented in the way shown in Figure 5.1. At the top of the stack lies the reasoning systems BN and CBR, implemented using GeNIe and jColibri respectively. Since jColibri is implemented in the programming language Java, this is naturally the chosen language for the rest of the system as well. The arrows between the two top layers signify the work done by the wrapper class for the BN, jSmile, that provides methods for interacting with and updating the BN. Basically the wrapper classes comes with exposed methods that are implemented in a way such that the user do not need to worry about parsing the .XDSL documents which GeNIe constructs to represent the BN with textually.

When downloading jColibri it comes by default packaged with Hibernate as an ORM middleware, and the designer is encouraged to use it as well. Hibernate provides a way for object-oriented programming languages such as Java to map

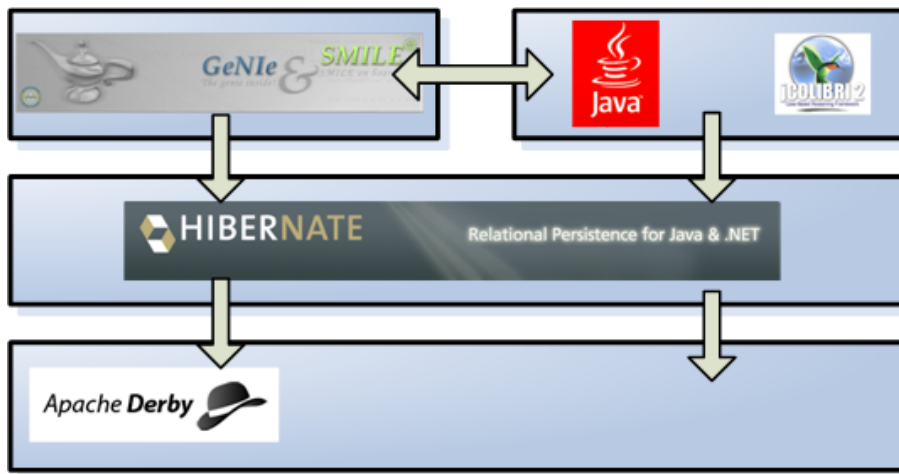


Figure 5.1: The technology stack of the system. The arrows reflect the communication between the various parts.

objects in the source-code directly to objects in the database without risking the integrity of the data. As a technology, Hibernate saves the designer a lot of trouble by handling a lot of details that are usually associated with database interactions. However, since one is passing control over from Java to Hibernate every time a database call is initiated it is harder to track the data flow, and if exceptions occur the debugging gets rather more complicated. All in all, some issues has been encountered with this approach, and will be further discussed in the Discussion section.

As a bottom layer we find the database, or case-base. jColibri is implemented with support for HSQLDB databases, which basically takes hold of some of the internal memory in the computer it runs on and constructs a virtual database. In a prototypical system, this approach works pretty well, since the case-base need not be permanently located anywhere for the system to work. However, in systems where the database need to be running in the background consistently, or at least for longer stretches of time, a more fully functional database management system is preferred and often required. Our choice was to use Apache Derby after encouragement from Tore Bruland, which is one of the supervisors, both from the fact that it is what he was using for the system their experimenting with in relation to the TLCPC project, but also since it has some key advantages. Firstly, it is an open-source project by Apache released under their 2.0 license allowing it to be: *“Reused without any modification by any project”*, among other things. It also has the advantage of being a small sized application so that it is easy to ship with software without blowing the executable out of proportions. Lastly, the entire database is implemented in Java and provides an embedded JDBC driver

which makes it possible for us to embed the database in our source-code and pack it with so no additional downloads are necessary when running the application. In the actual development process, Derby makes it simple to use in Eclipse in that it provides a plug-in for launching it in the running project, circumventing the need for either a stand-alone application or a shell script to run.

jColibri is configured in such a way as it requires four stages to be implemented in the main class of the system. In our case this class is `MealRecommender.java`, and the four stages are implemented as the main methods. Each method has responsibility for a certain part of the process as is explained in the coming sections.

Configure

The `configure` method is the first one to be called and is responsible for preparing the system. This involves establishing a connection to the database and running the SQL script which builds the database tables and their relations. The tables are holding the 60 or so cases, complete with descriptions like the recipe title, features corresponding to the ones found in the ontology and the BN, and a solution consisting of a grape type and the specific name of the wine recommended.

Furthermore, jColibri requires a database connector to be properly configured. The connector is responsible for connecting the database content to the case-representation found in the system, so that the reasoning process knows which features in the case-base are which in the system. It does this through an XML document describing what classes and files are containing the description of the cases and solution.

Lastly, `configure` sets up the connection to the Ontology through *OntoBridge*. *OntoBridge* is a sub-project of jColibri and is a library that handles the definition and use of ontologies in the CBR system. An ontology is basically a set of concepts describing the world and the relations between each such concept. The use of an ontology in a reasoning system makes it possible to store more knowledge in the cases by giving each feature or concept additional meaning through a “place” in the world. Once the ontology is made and *OntoBridge* initiated, jColibri can use the database connector to link features in the case-base with instances in the ontology.

Pre-Cycle

The pre-cycle iterates through the case base and loads each case into the memory, while at the same time constructing `MealDescription` and `MealSolution` objects of

these, linking the concept descriptions to the classes handling each of them.

Cycle

The cycle method is the core of the system and this is where all the function of the system lies. The call is wrapped in a while-loop so that the system will be able to execute several consecutive queries without needing a reboot.

For each run of the cycle method the system starts with initiating the Bayesian Network and the class responsible for handling this. It then proceeds to present the user with a similarity dialog, allowing each user to decide both what kind of metrics to use, but also what weight to assign each feature. E.g., the user may select `OntDeepBasic` as metric and set the weight to 100, 75, 50, which effectively means that the type of food is a more important feature than accessories, which in turn is more important than the sauce type.

The similarity metrics and the weights are passed along to a function while the query is passed along to the BN and set as *evidence*. The values calculated and returned by the BN are placed in a hash map made up of `String`, `Double` pairs, where the key is the grape and the corresponding value is the probability. Effectively this makes the system able sort out the k grapes with the highest probability and use this to guide the search in the case-base, ruling out cases where the solution does not contain any of these five.

Using the metrics stored in the similarity function, the CBR part of the system evaluates each case accordingly and use a pretty standard k -NearestNeighbour function to retrieve the top k cases from the case-base. These are stored in a `Collection` of cases and passed to the solution dialog and then presented to the user. If the user has selected, e.g., 3 as the number of cases to retrieve, two cycle buttons in the dialog allows the user to cycle between the three cases in order to compare their solution.

Post-Cycle

Post-Cycle disconnects the connector to the database and effectively shuts down the system. If additional steps are needed, all clean-up may be performed in this method.

5.3 The Explanation Generation

The classes responsible for explanation generation has been designed as an add-on to *Bacchus*. This approach ensured the system's availability for both our projects needs so that no restraints are posed on one or the other. The explanation part takes the form of two classes, one dialog class which presents the GUI and handles input, while the underlying class provides three distinct methods for generating explanations using either CBR, BN or a combination of the two.

The CBR explanation shows the user the similarity calculated between the query and the solution case. As it is implemented now, the explanation calls the class responsible for calculating this as the button is pressed. In other words, the computational operation takes place twice, once when the retrieve step is performed, and once when an explanation is requested. Since the number of cases used is so small, usually three or five, this approach does not really pose any significant delays, and the explanations appear almost immediately.

The similarity metrics are calculated by using the `OntDeepBasic` function from Section 4.5 for the two concept based features, and a simple `String.equal` function for the sauce. The latter returns 1 if the sauce is the same as the one used in the recipe and 0 if it is not. At this point one could benefit from using the custom similarity table described, but that particular metric was available too late into the project.

The same way of calculating the metrics are applied to the BN explanation generator, which is also called when the button is pressed. The operation for this part is a bit bigger, as it needs to iterate through the XDSL sheet produced by GeNIe containing all the information regarding the BN to find the right probabilities, as well as sorting them and writing them out. But again, the complexity of the system is not that great so the time delay is almost nil. However, in a larger system with lots of nodes and dependencies, this step should probably be taken care of at the same time as the other reasoning processes.

The probabilities which are presented are initially normalised to 1 in the network, but converted into a percentage format and given with at most two decimals for easier interpretation. Furthermore, the similarities are divided into three sections following the causal structure of the network. As the evidence is given by the user this is not presented, so the first layer is the intermediary food property nodes. These describes for the user what level of bitterness, flavor and freshness the food is likely to have. Following, a presentation of the requirements the wine should fulfill is given. These probabilities (also represented in the format of a percent) tells the user how likely it is that these particular characteristics will give a grape,

and thereby a wine, that fits well with the meal.

Lastly the user is given the k top grapes returned as a solution from the BN, reflecting the constraints put on the system. This allows the user to evaluate the result obtained by comparing the characteristics of the food and the requirements of the grape with the suggestions.

Chapter 6

Results and Discussion

The result of this project is two-fold. One is the implementation of the system, *Bacchus*, itself and the design for a combined CBR/BN reasoning system. The other is the explanation generation and presentation of the results produced by the system. Since much of the work on the knowledge-base has been done by Anne-Marit Gravem, my co-developer, and since her thesis was to explore how such a hybrid system might be implemented, the test results from this part of the project is in large part her design. However, both since these results forms the basis for explanations, but also since it was a co-effort, it is appropriate to present and discuss them here as well. What is not found here, though, is a detailed summary of all the testing done and the results obtained for every part of the system. For those interested in the fine-grained details, I advise you, no, encourage you to have a look at Anne-Marit Gravem's master thesis too.

The second part of this chapter is dedicated to the results from the explanation module of the system. A combination of the pragmatic results of testing as well as the more method-based results is presented and discussed.

In the last part of the chapter a general discussion regarding the progress of the project will be given. At the end, the future work that needs to be done in order to make *Bacchus* an invaluable tool for the novice wine-enthusiast is discussed. There will also be drawn parallels between the principles behind this particular system and the one aimed at palliative care.

6.1 Results

As stated in the introduction (Section 1.1) the goal for this thesis was to implement a prototype system in order to explore how the two reasoning paradigms of CBR and BN could be combined and how to produce good explanations from this combination. The result achieved from implementing a prototypical system contains two parts, the system itself and the explanation module. In the coming sections the results from the system will be presented and the explanations given will be compared to the explanation goals and categories presented in Chapter 3.

Two methods of testing the system were completed. In the first we tested the accuracy of the BN isolated from the rest of the system. This was done to ensure that the network was able to provide the correct grapes most of the time, if not it would impact the system as a whole. In the test setup for the system, 13 courses were constructed and run through the system. An example of one of these is found in Table 6.1a. The results from the BN was compared to information in a booklet called “*Nyttig om mat og vin*”, which is a small book produced and distributed by Vinmonopolet that lists suggestions for wines for over 100 meals. Along with this, information from their web page was also used. The initial test showed us that the BN managed to return at least one correct grape in 8 of the 13 courses, which in itself is a decent result. The problem found in this particular test was that the grape probabilities were clustered in the 9-13% range, which essentially mean that the difference between them are almost non-existing. A revision of the BN was performed where the CPT were re-filled with values. Before the revision we had normalised the values in each grape, meaning that the CPT for each grape summed up to 100%. In the new CPTs we normalised over each of the characteristics instead, giving grapes which had many of a particular combination a higher probability of being the correct one than one that had fewer. E.g. if the characteristics of the meal required TanningAgents=Good, Richness=Good and Fruitiness=Good and *Barbera* had 10 of 20 bottles matching this, the probability for *Barbera* would be 50%.

$$P(\text{GrapeCharacteristics}) = \frac{\text{grape's \# of bottles with this characteristic}}{\text{total \# of bottles with this characteristic}} \quad (6.1)$$

The equation above shows the function for this new property of the BN. The result of the change, however, was not as great as one could hope. Thus, in order to further discriminate between the grapes a log-odds-ratio calculation was performed on all the grapes. Basically this function calculates the ratio between probabilities

in order to see the relative probability for an event to happen comparing groups. In our case we were interested to see which grapes that was less or more probable when the evidence had been set. The function for the log-odds-ratio is:

$$L(O) = \log\left(\frac{P(\text{grape}|\text{evidence})}{P(\text{grape})}\right) \quad (6.2)$$

The result of these revisions of the BN was that the in 5 of 13 cases the network provided at least one correct grape compared to the recommendations from our source. This particular result was not all that positive as it was a worse performance than before revising the functions. However, when consulting our expert to discuss the results she stated that it was not possible, on the basis of the information provided, to disregard the grapes that were “wrong” according to the recommendations. There were two particular reasons for this: Each of the grapes found could possibly be a good match since it is a lot more to wine and food matching than the type of grape used, e.g. the combination of grape type, region, growth conditions etc. In addition to this, the recommendations from Vinmonopolet are not the one true answer, they are great matches but they are often picked among the more popular grape types since that is what people wants. Thus, a less familiar grape, like CabernetFranc, might have a perfect wine, but might not be recommended since it is not as well known. However, the poor discrimination shows clearly that the model is too general as it is now, and need more properties. Changes like this is discussed in detail in the Future Work section (6.2).

In addition to testing the BN, the whole system has been thoroughly tested by Gravem. A short summary of the method applied and the results found will be presented here, but details will be omitted where it does not affect the comprehension.

Table (6.1a) has been provided by Gravem as this testing is not something I have done.

For the whole system’s test the same 13 courses were used as test-input to the system. An example of such a case and the 5 retrieved solutions is given in Table 6.1a. Basically, all five wines presented in the solution cases are based one of the five grapes shown Table 6.1c, which has the characteristics found to be required by the BN. The particular wine that is recommended, where the name and vintage is given in the middle column, is based on recommendations by Apéritif. The similarity listed at the left column is the case-based similarity between the query and the case in the *Used in course* column. The information given in Table 6.1b is what each of the solution’s meals contained, and would allow the user to make up her own mind whether or not the meal is sufficiently alike.

Course 3 - Beef with baked potatoes and Bearnaise			
#	Similarity	Wine Suggestion	Used in course
1	0.68	Domaine de l'Ameillaud Cairanne 2006	Beef with asparagus, mushrooms and potato puree
2	0.51	Elsa Malbec 2005	Beef Pebre
3	0.45	Vidal-Fleury Crozes-Hermitage 2006	Reindeer fillet
4	0.43	Ch. Franc Cardinal 2002	Beef with aroma butter and potato salad
5	0.42	Lindemans Cawarra Shiraz Cabernet 2007	Reindeer stew with cheese sauce

(a) The case results.

#	Meal contents
1	BeefTenderloin, Mushroom, Asparagus, MashedPotatoes, CreamSauce
2	BeefTenderloin, Mushroom, Rice, NoSauce
3	Reindeer, Mushroom, BakedPotatoes, GameSauce
4	BeefSirloin, Salad, BoiledPotatoes, NoSauce
5	Reindeer, Mushroom, BoiledPotatoes, Vegetable, CreamSauce

(b) The contents of the results.

#	Grapes
1	Carignan, Syrah, Grenache
2	Malbec
3	Syrah
4	Merlot, CabernetFranc
5	Syrah, CabernetSauvignon

(c) The grapes present in the wines.

Table 6.1: Test results course 3

These last tables were brought down to our expert at Vinmonopolet, Jorunn Hoøen, for her to give an expert opinion regarding whether the proposed wines would be a good fit or not. We presented them to her one-by-one and asked if the wines proposed was a good match to the query meal. Basically she had three replies: Not a good match because[...], an OK match, a good match. The OK response meant that she herself would have picked something different, but that the wine would work just fine.

Considering the fact that neither Gravem nor me are domain experts we received some surprising results. In 12 of the 13 solutions that we presented Jorunn with, we received either that they all were OK or good matches, and most of them were not merely OK, but good. The one case which did not receive good judgment was the first of the 13 cases. The main-ingredient there was chicken-fillet, while the solutions suggested were recipes for e.g. elk-stew and beef-stew. Obviously, a wine which is recommended for elk-stew, which is a dish containing a sauce with much cream, is a bad choice for a chicken based dish which need a lighter and more juicy wine. After some discussion, however, we realised that the test-setup for this particular course was wrong. Where there was supposed to be *NoSauce*, there was *GravySauce* instead. Because of this the system had retrieved “wrong” cases. Since this, in effect, negates the value of that particular test we chose to leave it out of further evaluation. When leaving this out we are left with 12 tested courses. And, surprisingly enough, the system achieved an astounding 87% accuracy on the testing. That means that 52 out of 60 proposed wines were deemed a good fit by our expert.

However, the one test which was left out makes an obvious flaw in the system apparent. The fact that choosing the wrong kind of sauce to go with the meal makes such a big impact indicates that the either the BN or the CBR process needs revising. After analysing the results it seems like the CBR part of the system is the culprit. Basically, since its global similarity function is `Average()` each of the three ingredients in the course are given equal amount of weight. This may be compensated for by the user if she choose to adjust the slider representing the weight of each attribute in the similarity dialog. However, if all are equal, as they tend to be, the system should somehow compensate for this by default, as the main ingredient is the most important part in choosing a wine. This particular problem might solve itself, however, if more features were introduced than just the three used today. An average of three is a pretty coarse metric to use, and providing more features to compare should increase the value of the comparison as a whole by decreasing the impact of a single feature. Ways to do this is discussed in Future Work (Section 6.2).

In Figure 6.1 a screenshot of how the explanation from the system is presented to

Description

FoodType: ChickenFillet
 Accessories: [Salad]
 Sauce: NoSauce

Solution

RecipeTitle:
 WineTitle: Ravenswood Zen of Zin 2004
 WineOrigin: California
 Grapes: [Zinfandel]
 Similarity: 0.7777777777777778

The similarity between ChickenFillet and ChickenFillet is 1.0.
 Combined with the chosen accessory [Pasta] which match with the proposed meal's [Salad] with similarity 0.3333333333333333
 The sauce has similarity null.
 The mean of these values give the similarity stated and is the most similar case in the case-base.

The meal-ingredients chosen is likely to result in the following characteristics:
 Bitterness: Low=100%
 Freshness: Medium=60%
 Flavor: Bland=80%

Explanation: Based on the meal's characteristics, the characteristics of the wine should meet the following requirements:
 Richness: Good=59.4%
 Freshness: Low=40%
 TanningAgents: Good=51%

The grapes most likely to meet these are:
 Zinfandel=0.67%
 TintaMiuda=0.38%
 Montepulciano=2.72%

Figure 6.1: A screenshot of the combined explanation dialog as presented in the system.

the user is shown. The three buttons, CBR, Combination, and BN will present the user with different parts of the explanation, while the last button closes the window and takes the user back to the result dialog.

As presented in 3.1 three of the five explanation goals discussed in (Sørmo et al., 2005) are deemed important to this system as well. The three; *Transparency*, *Justification*, and *Conceptualization*, are important to, if not fulfill, at least take into consideration when designing this system. To recapitulate the key points of the goals: Transparency shows the user the reasoning process behind the solution, Justification justifies the answer based on e.g. similarities found, and Conceptualization explains expressions and words to make the explanation comprehensible.

In *Bacchus* the user basically has all the control when it comes to input into the system. It is explicitly shown and, except for the default configuration, required for the user to actively engage in deciding how the system shall compare the query against the case-base. Giving the user this sort of information and choices not only allow her to reflect over and view the difference an alternative setup gives, but also give, at least indirectly, an explanation for how the system acts in order to produce the solution. This point goes some way into fulfilling the transparency issue. Some way since this information only represents the CBR part of the system, the BN is still hidden from the user at this stage. It is not until, and if, the user choose to

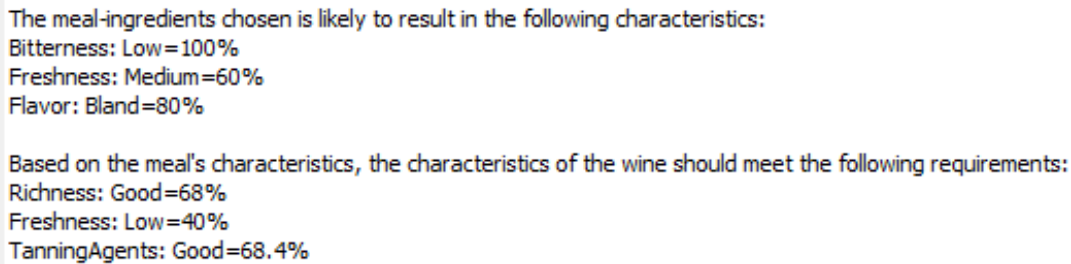
view the explanation for the solution that the BN will show its reasoning steps, and even then only indirectly since the steps are not explicitly connected to show the relation between two nodes. One option here is to give the user a graphical overview of the BN, complete with nodes and links, so that the actual model used in the reasoning is easily available. In small domains such as in *Bacchus* one would not necessarily need to do anything other than show the network, in larger domains, however, it would be feasible to color the links which are activated during the process to more clearly present the relation between the nodes and the result.

What the BN fails to show in transparency is somewhat gained in justification, as the system presents the user with the probabilities extracted from the probability tables in the network. Since each of the grape nodes are influenced by the same parent nodes, an overview of both the target's parent nodes and their parents again will show the user how the system reached the conclusion that the given grape(s) was/were the ones with the highest probability. The values presented are thus all the information used by the system and will justify the solution based on this. If the user feels the justification is less than adequate, it is at least not because of reasons unknown, but rather that she disagrees with the knowledge model.

When the solution was tested on our domain-expert, Jorunn Hoøen, the explanation the system gives was shown and described to her at the same time, after giving her a run through the system. As we had already discussed the BN with her, she had a general idea about how the different parts of the network was connected.

Her response to the explanations shown was unconditionally positive. Specifically, she was surprised at the possibility to view this kind of information and thought that what she could get out of it was a good indication for the reasoning that had been done. As an explanation for this reaction, a brief look at what her reasoning processes was like when deciding which wine that would fit is in order. Basically, she would look at what the meal consisted of in terms of both the type of main ingredient (meat, fish, game etc.) and then the accessories and sauce. When confronted with e.g. *MashedPotatoes* she reasoned that this type of potatoes usually contain some amount of butter and cream, making the wine more rich (“[...]which elevates the richness of the wine”), and because of this the wine would require a larger amount of freshness than otherwise. Same with e.g. pork rib, where the meat has a defined line of fat, which also makes the meat fat. Because of this the wine needs tanning agents and freshness to complement the food.

In Figure 6.2 the BN explanations for the intermediary nodes is shown. When looking at the explanations presented, and then especially the BN part of the explanation (which is the one which describes the food-wine dependencies), the kind of reasoning explanation provided is more or less the same as what she



The meal-ingredients chosen is likely to result in the following characteristics:
Bitterness: Low=100%
Freshness: Medium=60%
Flavor: Bland=80%

Based on the meal's characteristics, the characteristics of the wine should meet the following requirements:
Richness: Good=68%
Freshness: Low=40%
TanningAgents: Good=68.4%

Figure 6.2: An enlargement of the explanation panel, showing the probabilities extracted from the BN from the intermediary nodes

was doing in her head. The difference is that it is done through probabilistic reasoning instead of years of accumulated knowledge and experience as it is in her case. In particular, the representation of the food characteristics together with the characteristics which the wine chosen should fulfill, gave a good pointer as to what the wine should be like she said. Another, more subtle, part which she commented on were the use of the standard description of wine property (Richness, TanningAgents, Fruitiness), and she applauded the system for using established standards with which to evaluate the wine. This particular trait of *Bacchus* was something intentionally done in order to have a reliable source of knowledge to use in our model, and might be something that could have been emphasised in the explanations. As it is now, one would need to notice the likeness in the names, while a small illustration, like the pie-charts used at Vinmonopolet, of what each value meant could've helped with this.

On the other hand, the CBR part of the explanation didn't seem to hold the same interest to her. The reasons for this might be many. The first of which is that the BN explanation was emphasised first, and that part seemed to give her the information needed. A possible second reason is that this part only gives an explanation of *how* the solution case is found to be so-and-so equal to the query. During the discussion of the system's test results, the focus lay solely on the match between the presented solution's wine and the meal from the query, and not how the case solution was equal and thus presented since that is a little out of her domain. Although she did not give any outward signs or comments about it, it might simply have been because she took that part for granted. To clarify: The way the system shows the solution, by presenting the whole solution case to the user, is by many regarded as an explanation process in itself. The emphasis on this is removed when presenting the explanations in the way the system does, by dividing the solution representation and the rest of the explanations. We have done it like this in *Bacchus* since the system is used by two separate projects, but essentially all

the information could have been provided in the same screen and thus equalised the two. The effect of this could have been interesting to examine, however, because of time-issues a larger test group was not used. But, if we use the reaction to the solution screen (Figure 4.9) as a reaction to the case-as-explanation approach for analysis she had a few comments. Basically it was that the representation of the wine could have been better, i.e. more information about what region it is from etc. Except for this she quickly concluded that the presented solution would fit the meal, and that the query and the solution were alike in key areas which influence the choice of wine (e.g. fat, dairy products etc.).

The combination of the two both enables transparency of the reasoning and justification of the solution. However, since this is merely a prototype of a system, there are many parts of the system which could have been explained which currently are not. These will be further discussed in the Future Work section (6.2).

Conceptualisation is not currently implemented, but is something that *Bacchus* could have benefited from a great deal. Obviously, as a wine recommendation system one can not expect the users to have any knowledge of the terminology used in the system, and therefore a clarification of key words could have been a nice addition. One option could, as noted in (Mao and Benbasat, 2000), be to include some sort of hyperlinking for the words that are specific to either the domain, for novice users, or for words that are specific for the technical implementation, like e.g. “ontology”. Or, of course, for both options.

Table 3.2 gave an overview of the properties from (Lacave and Díez J, 2002) which are included in *Bacchus*. The combination of the explanations provided in the system aims at giving a description of the reasoning process at the macro-level, with a causal model as the basis. The CBR based explanation falls a bit outside of this categorisation scheme as it is intended for networks, however in large part it does conform to the description of the reasoning as well. But because of the lack of a revise and retain part in the system, complex explanation generation based on previous experiences is hard to accomplish. For an explanation to be able to compare and present convincing explanations based on something other than the similarity measured, learning from successes and failures is an important part. This also affects the types of explanations that can be provided, e.g. explanation of anomalies or explanations in the form of earlier successes relies on data not available in *Bacchus*. In the domain of health science it is obvious that classification of solved cases is important, and this kind of functionality and what it would provide and/or demand of the explanation capabilities of the system should be further investigated. However, as said before, because of time-constraints, a full CBR cycle implementation was beyond the scope of this project.

6.1.1 Discussion of the Progress

The actual implementation of the system is in much part Anne-Marit Gravem's. Her thesis's task is to examine how to best combine and integrate the two paradigms, and as our two tasks coincide in a large degree, mine is but an extension of the functionality, we have co-operated for the final stages of the implementation.

At first we developed each our system in parallel, which efficiency wise is not really the best option. The system originally designed for this thesis was aborted after a few weeks of implementation when we realized that we were doing the exact same things. At the point where it was discarded, the system had a functioning case-base with the retrieval step implemented, but not yet connected to the BN. The actual BN connection is in large part done through a wrapper class called `jSmile`, so effort wise the two projects wasn't all that far apart.

The decision to merge our systems has of course saved me some work, as Gravem had spent a lot of time collecting and inserting data into the case-base, which was something I had circumvented in my system as I operated with a downloaded data-set inserted through a custom Python script. Over the weeks where we co-operated on the system my contribution were help with tweaking and some bug-fixing, as well as further implementation where necessary. In addition I had a part in the extension of the BN, as well as constructing the probability tables. As well as saving me some work, some extra time was spent on familiarising myself with the code provided by Gravem.

As in most software projects, the actual time to implement the system went far beyond what any of us expected. The actual code for the system is not all that big, but because everything is based on third party framework and software a lot of time in the initial phases of implementation went into getting to know these tools. `jColibri` offered a neat tutorial example which quickly made us come up to speed with how the framework was designed. The trouble started when the system was integrated with, initially, the HSQL database where some unknown bug made *Bacchus* unable to connect properly. Apache Derby solved that particular problem, but then the problems with having a middleware such as Hibernate became apparent. As noted earlier, when handing control over from Java to Hibernate, a lot of the exception handling inherent in Java is no longer available. This leads to a frustrating time of debugging, trying to locate the most probable source of the problem and fixing this. If that does not work, one is forced to repeat the process. E.g., in one of the XML documents handling the database configuration and mappings the author had accidentally inserted an extra space in the name of a class file, making Hibernate unable to locate that particular file. It was not until after the whole file had been rebuild that the problem was solved.

In hindsight the approach for making the explanation part of the system might not have been optimal. In essence, implementation of the core system took a lot more time than anticipated, leading to much too short time on implementing the explanations. In addition, since the system use jSmile to interact with the BN, effectively making the BN an external part of the system, the methods available for pulling data out of the BN is hidden. This leads to an implement-test-redo kind of method where different approaches has to be tried. Each time the data returned is not the correct, a new approach must be implemented. A better approach might have been to only superficially include a reasoning part, i.e., simulate the system without actually implementing it. This way, many more types of explanations could have been tested, including different ways of presenting them to the user. An approach such as that, though, could potentially lead to erroneous results since one would have to make assumptions about what kind of information was available from the system, as well as what kind of response the system would give in certain situations.

These kind of problems could have been, at least partially, eliminated by implementing the system from scratch. That being said, and despite the problems encountered, such an approach would be a too large a task for us to have accomplished. It might be worth considering for future systems though, since control of the reasoning parts is to a large extent what the explanation designer need if one was to go from a knowledge-light to a more knowledge-intensive approach.

6.2 Future Work

While working on and testing the system, a lot of potential has been discovered in how to make the system even better than it is today. Some of the points here comes from discussion with Aamodt and Gravem, and some are a result of response and discussion with Jorunn Hoøen.

Basically, what the system needs in order to become a full fledged recommender system in the meal-wine domain is more and better data to base its reasoning on. *Bacchus* suffers under the fact that neither Anne-Marit nor me are experts within the domain, and as such many key points are missing. E.g. in the BN the three nodes representing the food characteristics is not enough. In order to catch all the nuances in the food needed to accurately propose a solution, tables for nodes such as; the content of dairy products, salt, and spices should be addressed, as well as the cooking preparation made for accessories and meat (boiled and fried meat is two different things). These are but a few examples, but ideally, experts from both the food and the wine domain should be consulted to get the dependencies right.

This way one would at the same time get the characteristics of the food better, since as it is now it is based on our *beliefs*.

In addition, the network do not offer any way to differentiate between wines based on the unique conditions in which each of them has been produced. A large part of what decides how a wine tastes is the specific conditions under which it grows (i.e. terroir) and maturing properties (e.g. oak barrels for 3 years). For instance, a Riesling tastes different from a Chablis because of the difference of sun, temperature, soil etc., while a *Reserva* wine made from Sangiovese is more round in taste than a Sangiovese based wine with shorter maturing period. These and other properties of the wine is what makes that particular bottle especially good for that particular combination of food, and not just the grape. However, if one were to include all these different influences into the BN one would need two things that we didn't. The first is knowledge, i.e. access to an expert or the equivalent information about how these things are connected, and secondly time to properly implement a more complex network.

The case-base and the ontology needs to be expanded as well. It should be more focused on the specific courses, and not just the general similarity between them. E.g., the entry *Chicken* should lead to a result mainly consisting of several different types of dishes with chicken as the main ingredient, but maybe with different accessories and sauces. As it is now, the system performs well enough, but without a stronger link between the meat that is equal some solutions will obviously not be optimal. This sort of a more specialisation of the CBR part of the system might also lead to a better form of explanation, since it would be easier to classify results as either sure-fire choices (of wine) or as something a bit more risky and justify the retrieval based on this.

Specifically for the explanations some work needs to be done: What is demonstrated through *Bacchus* is a way to give the user a description about what happens behind the scenes, transparency, as well as a partial justification. Partial because it does no explicit work on justifying the chosen case. The justification done is based solely on the similarity metrics, and implicitly by comparison to other cases. Studies suggest that humans respond well to being presented opposites at this stage, i.e. showing the user a similar case but where the solution did not lead to success. However, to be able to implement this in *Bacchus* we would need to implement some form of the revise and retain step, able to classify cases and index them based on this. As it is now, all cases are (un)classified as successes, and no discrimination is possible. What does work well, though, is the strong connection between the causality in the model and the causal reasoning employed by the expert, and is something which could benefit greatly from being expanded upon more.

To further increase the potential a stronger link between the cases and the model should be introduced. For example could wine characteristics be stored both in the cases and in the BN, either in addition to or instead of the grapes. This way the system could reason about specific properties of the wine in the BN and use the same information as an index in the case-base, thus getting better retrieval results.

Chapter 7

Conclusion

The report has described how the combination of two reasoning paradigms might pave the road for better explanations in the future. By starting out with the basics within the field and the hypothesis for why this might be done better, the intention behind the design of *Bacchus* has hopefully been made clearer.

What has been demonstrated in this report is basically how and why the combination of CBR and BN can work well in a recommender system. In health science it is important to have a strong foundation for every action taken toward treatment of patients, as it is their health and possibly their lives which are at stake. Many successful decision support systems and recommendation systems are already in wide-spread use today, but none of them utilise the full strength of a combination of two paradigms, and therefore we believe that *Bacchus* has something to offer.

The strength of the system comes from the fact that both previous experiences and statistical data work together to provide the best possible solution. Where CBR handles certainty and facts collected from expert sources, the BN deals with uncertainty and beliefs. The first is a good representation of the world as it *have* been experienced already, while the latter provides data about the rest of the world, the part which is mostly *unknown*, going a long way to reflect the way humans perceive our surroundings.

Even though the idea behind the design is good, the implementation presented still needs further work and experimentation in order to be viable for use in larger and more complex domains. The solution presented shows one option in how to give the user full access to and insight into the system and its processes. The method of modeling the world in a causal network is not a new one, neither is providing solutions in the form of old cases. The combination of the two however, might be a new one, and as the results show, the system is capable of achieving good results

even with much uncertainty in its model. The presentation of the reasoning process as a causal chain of data which influence each-other conformed, in our case, to the reasoning done by the domain expert. Along with the human-reasoning based method of CBR, the combination of these two paradigms will with more work lead to powerful explanation facilities in decision support systems. However, it would be interesting to try to combine the BN explanations with more complex CBR based explanations as this is not something done in this report. The possibility for a more knowledge-intensive solution like a semantic net representation of the cases together with the causal properties of the BN should possibly lead to some interesting solutions, which in the end would only benefit the users of these decision support systems.

References

- Michael A. Nakao and Seymour Axelrod. Numbers are better than words. *The American Journal of Medicine*, 74:1061–1065, 1983.
- Agnar Aamodt. Knowledge-intensive case-based reasoning in creek. In *Advances in Case-Based Reasoning*, volume 3155/2004, pages 793–850. Springer Berlin / Heidelberg, 2004.
- Agnar Aamodt and Enric Plaza. Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1): 39–59, March 1994.
- E. Armengol, A. Palaudaries, and E. Plaza. Individual prognosis of diabetes long-term risks: A cbr approach. *Unpublished as of 2000*, 2000.
- E. Ray Bareiss, Bruce W. Porter, and Craig C. Wier. Protos: an exemplar-based learning apprentice. *International Journal of Man-Machine Studies*, 29(5):549 – 561, 1988. ISSN 0020-7373. doi: DOI:10.1016/S0020-7373(88)80012-9.
- Juan José Bello-Tomás, Pedro A. Gonzalez-Calero, and Belen Diaz-Agudo. Jcolibri: An object-oriented framework for building cbr systems. In *Advances in Case-Based Reasoning*, pages 32–46. Springer Berlin / Heidelberg, November 2004.
- Anna Bergström, Paola Pisani, Vanessa Tenet, Alicja Wolk, and Hans-Olov Adami. Overweight as an avoidable cause of cancer in europe. *International Journal of Cancer*, 91(3):421–430, 2001. Department of Medical Epidemiology, Karolinska Institute, Stockholm, Sweden; International Agency for Research on Cancer, Lyon, France; Department of Epidemiology and Center for Cancer Prevention, Harvard School of Public Health, Boston, MA, USA.
- Derek Bridge and Lisa Cummins. Knowledge lite explanation oriented retrieval. *Computing and Informatics*, 25(2-3):173–193, 2006.

- B. Chandrasekaran, M.C. Tanner, and J.R. Josephson. Explaining control strategies in problem solving. *IEEE Expert*, 4(1):9–15, 1989.
- 1971 Cortez, Paulo, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. 2009.
- Donal Doyle, Padraig Cunningham, and Paul Walsh. An evaluation of the usefulness of explanation in a cbr system for decision support in bronchiolitis treatment. *”Computational Intelligence”*, 22(3-4):269–281, 2006.
- Chris Elsaesser and Max Henrion. Verbal expressions for probability updates: How much more probable is ”much more probable”? In *UAI ’89: Proceedings of the Fifth Annual Conference on Uncertainty in Artificial Intelligence*, pages 319–330, Amsterdam, The Netherlands, The Netherlands, 1990. North-Holland Publishing Co. ISBN 0-444-88738-5.
- Shirley Gregor and Izak Benbasat. Explanations from intelligent systems: Theoretical foundations and implications for practice. *MIS Quarterly*, 23(4):497–530, December 1999.
- Jonathan L. Herlocker, Joseph A. Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *CSCW ’00: Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, New York, NY, USA, 2000. ACM. ISBN 1-58113-222-0. doi: <http://doi.acm.org/10.1145/358916.358995>.
- Janet Kolodner. *Case-based reasoning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993. ISBN 1-55860-237-2.
- Phyllis Koton. Reasoning about evidence in causal explanations. *AAAI-88 Proceedings*, pages 256–261, 1988.
- Carmen Lacave and Francisco Díez J. A review of explanation methods for bayesian networks. *The Knowledge Engineering Review*, 17(2):107–127, 2002.
- David B. Leake. An indexing vocabulary for case-based explanation. *AAAI-91 Proceedings*, pages 10–15, 1991.
- P Lichtenstein, NV Holm, PK Verkasalo, A Iliadou, J Kaprio, M Koskenvuo, E Pukkala, A Skyttte, and K Hemminki. Environmental and heritable factors in the causation of cancer—analyses of cohorts of twins from sweden, denmark, and finland. *The New England Journal of Medicine*, 343(2):78–85, July 2000.

- Ji-Ye Mao and Izak Benbasat. The use of explanations in knowledge-based systems: Cognitive perspectives and a process-tracing analysis. *J. Manage. Inf. Syst.*, 17(2):153–179, 2000. ISSN 0742-1222.
- David McSherry. Explaining the pros and cons of conclusions in cbr. In *Advances in Case-Based Reasoning*, volume 3155/2004, pages 149–165. Springer Berlin / Heidelberg, 2004a.
- David McSherry. Incremental relaxation of unsuccessful queries. In *Advances in Case-Based Reasoning*, pages 131–148. Springer Berlin / Heidelberg, 2004b.
- David McSherry. Explanation in recommender systems. *Artificial Intelligence Review*, 24:179–197, 2005.
- Conor Nugent, Dónal Doyle, and Pádraig Cunningham. Gaining insight through case-based explanation. Technical Report UCD-CSI-2007-12, University College Dublin, 2007.
- Christopher K. Riesbeck and Roger C. Schank. *Inside case-based reasoning*. Artificial Intelligence. Routledge, illustrated edition, 1989.
- Roger C. Schank and David B. Leake. Creativity and learning in a case-based explainer. *Artificial Intelligence*, 40(1-3):353 – 385, 1989. ISSN 0004-3702. doi: DOI:10.1016/0004-3702(89)90053-2.
- Edward H. Shortliffe, Randall Davis, Stanton G. Axline, Bruce G. Buchanan, C.Cordell Green, and Stanley N. Cohen. Computer-based consultations in clinical therapeutics: Explanation and rule acquisition capabilities of the mycin system. *Computers and Biomedical Research*, 8(4):303 – 320, 1975. ISSN 0010-4809. doi: DOI:10.1016/0010-4809(75)90009-9.
- AA Somogyi, DT Barratt, and JK Coller. Pharmacogenetic of opioids. *Clin Pharmacol Ther*, 81:429–444, 2007.
- Frode Sørmo, Jorg Cassens, and Agnar Aamodt. Explanation in case-based reasoning - perspectives and goals. *Artificial Intelligence Review*, 24:109–143, 2005.
- H.J Suermondt and Gregory F. Cooper. An evaluation of explanations of probabilistic inference. *Proc Annu Symp Comput Appl Med Care*, pages 579–585, 1992.
- W. Swartout and S. Smoliar. On making expert systems more like experts. *Expert Systems*, 4(3):196–207, 1987.

Randy L. Teach and Edward H. Shortliffe. An analysis of physician attitudes regarding computer-based clinical consultation systems. *Computers and Biomedical Research*, 14(6):542 – 558, 1981. ISSN 0010-4809. doi: DOI:10.1016/0010-4809(81)90012-4.