Knowledge-Intensive Case-Based Reasoning in CREEK

Agnar Aamodt

Department of Computer and Information Science
Norwegian University of Science and Technology (NTNU)
NO-7491 Trondheim
Norway
agnar.aamodt@idi.ntnu.no

Abstract. Knowledge-intensive CBR assumes that cases are enriched with general domain knowledge. In CREEK, there is a very strong coupling between cases and general domain knowledge, in that cases are embedded within a general domain model. This increases the knowledge-intensiveness of the cases themselves. A knowledge-intensive CBR method calls for powerful knowledge acquisition and modeling techniques, as well as machine learning methods that take advantage of the general knowledge represented in the system. The focusing theme of the paper is on cases as knowledge within a knowledge-intensive CBR method. This is made concrete by relating it to the CREEK architecture and system, both in general terms, and through a set of example projects where various aspects of this theme have been studied.

1 Introduction

A knowledge-intensive case-based reasoning method assumes that cases, in some way or another, are enriched with explicit general domain knowledge [1,2]. The role of the general domain knowledge is to enable a CBR system to reason with semantic and pragmatic criteria, rather than purely syntactic ones. By making the general domain knowledge explicit, the case-based reasoner is able to interpret a current situation in a more flexible and contextual manner than if this knowledge is compiled into predefined similarity metrics or feature relevance weights. A knowledge-intensive CBR method calls for powerful knowledge acquisition and modeling techniques, as well as machine learning methods that take advantage of the general knowledge represented in the system.

In the CREEK system [3,4,5], there is a strong coupling between cases and general domain knowledge in that cases are submerged within a general domain model. This model is represented as a densely linked semantic network. Concepts are inter-related through multiple relation types, and each concept has many relations to other concepts. The network represents a model of that part of the real world which the system is to reason about, within which model-based reasoning methods are applied. From the view of case-specific knowledge, the knowledge-intensiveness of the cases themselves are also increased, i.e. the cases become more "knowledgeable", since their features are nodes in this semantic network.

The focusing theme of this paper is cases as knowledge within a knowledge-intensive CBR method. This will be made concrete by relating it to the CREEK

architecture and system, both in general terms, and through a set of example projects where various aspects of this theme have been studied. To give an initial hint at the main issue, Fig. 1 characterizes some aspects of CBR methods along what may be

Fig. 1. The knowledge-intensiveness dimension of CBR methods

called the knowledge-intensiveness dimension. The early nearest-neighbour-based methods are at the one end of the scale, while the CREEK system is illustrated closer the other end. Some typical characterizations of knowledge-intensive CBR methods (right part) and knowledge-empty or knowledge-lean methods (left part), are listed.

As Fig. 1 indicates, the notion of knowledge-intensiveness is not an either/or issue. CBR systems may be more or less knowledge-intensive. The meaning of the term "knowledge-intensive" may also vary, depending on what viewpoint to the concept of knowledge that an author or research group has. Further, when we look at the contents of a case, what some people refer to as knowledge may be referred to as information by others – or even as data. This is not surprising, since a data structure, such as a case, can serve several roles in a system. In order to get a better understanding of the concept of knowledge, as it is interpreted in CREEK, we will therefore start by clarifying what we see as the main distinction between knowledge, information, and data, related to the different roles a case may have. The next chapter defines the three terms from that perspective.

Explicit models of knowledge call for effective knowledge modeling methods and tools, both for manual model development and automated methods, i.e. machine learning. To support systems development within the CREEK architecture, some assumptions on the nature of knowledge modeling has been made, and an assisting tool has been developed to assist the knowledge modeling process. This is the topic of chapter 3. In chapter 4 the CREEK architecture and system is summarized, emphasizing knowledge content and how it is processed. Chapter 5 illustrates the architecture and system through a summary of recent and ongoing research projects. The paper is summarized and concluded in the final chapter.

2 What is knowledge in a CBR system?

There is, in general, no known way to distinguish knowledge from information or data on a purely representational basis. Attempts to make distinctions based on size or complexity are therefore likely to fail. Another option - and the one underlying the CREEK architecture - is to identify how and for what purpose the structures are used, i.e. what the various *roles* of data, information, and knowledge are in a case-based reasoning process. Their interpretation within the contexts they are applied, and by whom they are interpreted and applied, therefore become important. The latter aspect leads to the *frame of reference* problem of data, information, and knowledge [6], which is the problem of relating one of these entities to a subject of reference: Whose knowledge is it? For a discussion of this topic within the broader context of databases, information systems, and AI systems, see [7].

2.1 Data vs. information vs. knowledge

For any decision-making process, an environment is assumed in which a decision-making agent (i.e. a reasoning agent) receives input from and returns output to an environment external to it. In a simple set-up, the external environment is a user communicating through a terminal, and the decision-making agent is a terminal-based, advice-giving computer system. Within this context, the essential differences between data, information and knowledge are as follows (see Fig. 2).

Data are syntactic entities, i.e. uninterpreted characters, signals, patterns, and signs that have no meaning for the system (the subject of reference) concerned. Data are input to an interpretation process. Data become information after having been interpreted to give meaning. This is illustrated in Fig. 2 by the Data Interpretation arrow. Taking a human being as the subject of reference, a series of signals from a sensor, or the string "´Q)9§?8$%@*¨&/", is data to most of us, while "low interest rate", "increased blood pressure", and "the Gulf war" have meaning, and therefore are information. The meaning of these terms may be different for different systems (here: people), and it is each individual's knowledge about particular domains - and the world in general - that enable us to get meaning out of these data strings.

Information is interpreted data, i.e. data with meaning. It is the output from a data interpretation process, as just described. Once the data have been given an interpretation as information (an initial interpretation, at least), it is elaborated upon in order to be better understood, and in order to derive (infer) new information. This is illustrated by the Elaboration arrow in Fig. 2. Hence, information is input to this elaboration process, as well as output from it. The elaboration process is where the core decision-making processes take place. Often, in a real setting, elaboration and data interpretation processes are interleaved. Information is also the source of learning, i.e. the input to a learning process.

Knowledge is learned information, i.e. information that has been processed and incorporated into an agent's reasoning resources, and made ready for active use within a decision process. A widely shared view is that learning is the integration of new information into an existing body of knowledge, in a way that makes it potentially useful for later decision-making. New knowledge may also come from inference

processes within the knowledge body itself. This is illustrated by the vertical and the semi-circular Learning arrows in Fig.2, respectively. Knowledge, then, is the output of a learning process, after which it becomes the internal resource within an intelligent system that enables the system to interpret data to information, to elaborate and derive new information, as well as to learn more (the gray lines in the figure).

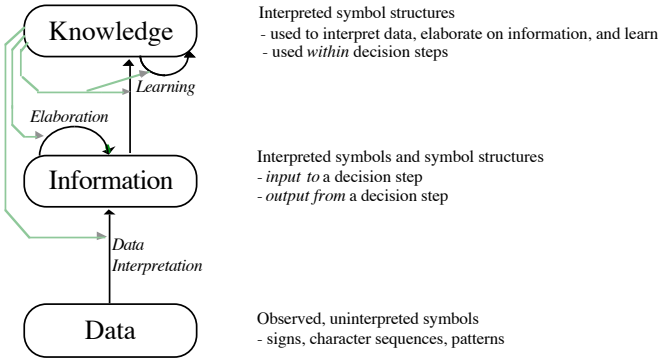


Fig. 2: The Data-Information-Knowledge model.

Note that the term knowledge is used here in a very general sense. It does not distinguish between 'true' and 'believed' knowledge. This is different from the influential branch of philosophy in which the term knowledge is used exclusively for statements that are true in the world, and where belief is used if truth cannot be ascertained (e.g. [8]). Other philosophical theories (e.g. [9]) have questioned this position, arguing that the logicist, or deductive-nomological philosophical view that lies behind that view is unable to explain major philosophical problems such as analogical reasoning, abduction, and scientific development.

2.2. Case roles in CBR systems

In Fig.1 some discriminating characteristics of knowledge-lean and knowledge-intensive methods were listed. CBR systems come in different shapes and fashions. From the above discussion, we see that in order for a system to reason, in the sense of interpreting data and deriving new information, it needs knowledge. Systems with no knowledge can do no reasoning in this sense. CBR systems that are placed at the left of the scale, will therefore typically be closer to information systems than knowledge-based systems. Note the peculiarity in that for an “information system”, as the term is commonly used, a human being is assumed to be the subject of reference, i.e. it is information for the human interpreter (and data for the system). In a knowledge-based system, however, knowledge as well as relevant parts of the information is with respect to the system. Below, the three main roles of cases in various systems, corresponding to their role as data, information, or knowledge, are highlighted, in order to contrast the CREEK approach with other approaches.

Cases as data for the computer system is the simplest mode, in which the system does not do case-based reasoning as such, but applies case-based methods for case indexing and retrieval. Since the system views cases as data only, it does not have knowledge of the items that describe case contents. The partial matching property of CBR is used to improve database retrieval by producing a ranked list of matching records rather than one exact match. The strength of computers as data managers and information handlers, where the frame of reference for information is the user, is combined with the strength of human beings for intelligent decision-making. Some types of help desk systems are examples.

Cases as information for the computer system implies that there is knowledge in the computer that is able to interpret and utilize case contents as information. If cases are information only – and not knowledge – the knowledge-based methods must be of some other kind, such as model-based or rule-based. The characteristic of a case-based system of this kind is that a substantial part of the system's information is organized as cases.

Cases as knowledge for the computer system, is the case-based *reasoning* approach per se, i.e. the case base is not merely a source of information for the user, but a knowledge base that is actively used in the system's reasoning processes. The full flexibility of viewing a case as data, information, and/or knowledge is therefore available. Cases may be the only type of knowledge in such a system or they may be combined with other knowledge types - as in CREEK. These systems exhibit learning in the full sense, since they incorporate new cases in a way that makes them immediately ready to be used in the solving of new problems.

From these different case roles, we see that a case-based system architecture can facilitate a gradual transformation from a pure database or information system, to a full-fledged knowledge-based system. In this way a system will always have its data available in a non-generalized form, and their active use can be incrementally put into effect by adding interpretation and reasoning capabilities to the system as the use of the system identifies what active decision support users really want.

3 Knowledge Modeling

Along with Clancey [10], a knowledge-based system can be viewed as a qualitative model of that part of the real world that the system is to reason about. Knowledge modeling, then, becomes the whole process that starts with a real world task environment, through several steps realizes a (partial) model of it in a computer system, and maintains that model over time. The knowledge of a system will to some extend be biased by the methods through which that knowledge was acquired and represented. A brief description of the high-level knowledge modeling framework underlying CREEK systems is therefore given.

The knowledge modeling approach is based on the combination of a top-down driven, initial knowledge acquisition process, and a bottom-up modeling process represented by continuous learning through retaining problem solving cases. The objective of the initial knowledge modeling task is to analyze the domain and task in question, to develop the conceptual, mediating models necessary for communication

within the development team, and to design and implement the initial operational and fielded version of the system. The knowledge maintenance task takes over where the initial knowledge modeling ends, and its objective is to ensure the refinement and updating of the knowledge model as the system is being regularly used. In Fig. 3 the two outer, rounded boxes illustrate these two top-level tasks of the knowledge modeling cycle. Within each of the two tasks, the major subtasks (rounded rectangles) and models (sharp rectangles) taken as input and returned as output from these tasks are shown. The modeling subtasks are indicated by their gray background.

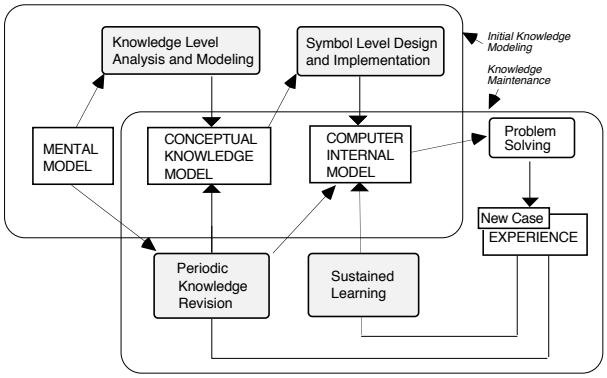


Fig. 3: The knowledge modeling cycle

Arrows indicate the main flow of knowledge and information, and show the most important input/output dependencies between subtasks and models. As shown by the area where the two large boxes overlap, the conceptual knowledge model and the computer internal model are shared by subtasks of both initial knowledge modeling and knowledge maintenance.

A knowledge modeling cycle typically starts with a high level specification (e.g. functional specification) of the target computer system, at some level of detail. The resulting submodels are structured into a conceptual knowledge model. The knowledge is described at the knowledge level [11,12], where the emphasis is to capture the goal-directed behavior of the system, and to model knowledge content from the perspective of the application domain, without being constrained by implementational limitations. A common starting point is to identify the main categories of the three knowledge types: *Task knowledge*, *Method knowledge*, and *Domain knowledge*. Task knowledge models what to do, usually in a task-subtask hierarchy. Tasks are defined by the goals that a system tries to achieve. Method knowledge describes how to do it, i.e. a method is a means to accomplish a task (e.g. to solve a problem). Domain knowledge is the knowledge about the world that a method needs to accomplish its task. Examples are facts, heuristics, causal relationships, multi-relational models, and – of course – specific cases (see [13] for a more elaborate discussion on knowledge level modeling for CBR systems). The conceptual knowledge model forms the basis for designing and implementing the computer internal model, i.e. the knowledge model of the operating target system.

This model is described at a level referred to as the symbol level, which deals not only with intentional knowledge content, but also with manipulation of symbols that represent knowledge in the computer.

The lower, partially overlapping box illustrates the main subtasks of knowledge maintenance. Knowledge maintenance starts when a system has been put into regular operation and use. The knowledge maintenance task has two optional subtasks as indicated in the figure. One is sustained learning, i.e. the direct updating of the computer internal model each time a new problem has been solved. The other is a periodic and more substantial revision process. As illustrated, this revision task may lead directly to the modification of the symbol level model (computer internal model), but it may also go through an update of the knowledge level model (conceptual knowledge model) first.

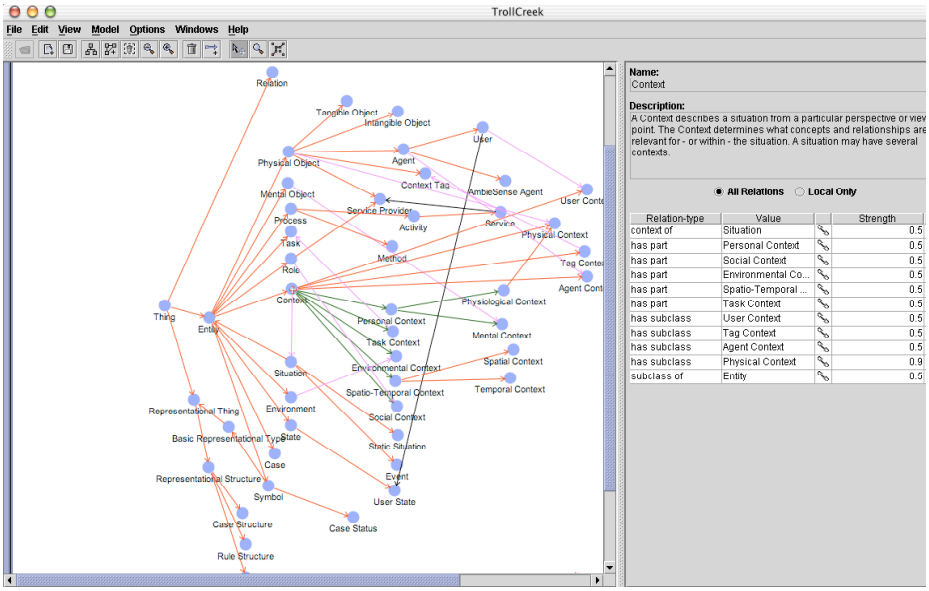


Fig. 4. The TrollCreek Knowledge Modeling Editor

To assist in the top-down modeling parts of the cycle described, a knowledge modeling editor is used (Fig. 4). A CREEK system comes with a top-level ontology, part of which is shown in the figure, from which the higher-level parts of the domain model is grown. Concepts, relations, as well as cases, can be constructed and manipulated in flexible manners through a knowledge map interface (to the left) or a frame interface (right part of the figure). The knowledge representation is the topic of the next chapter.

5 The CREEK system

The CREEK system is an architecture for knowledge-intensive case-based problem solving and learning, targeted at addressing problems in open and weak-theory domains [14]. CREEK contains several modules integrated within a common conceptual basis: The General Domain Model (see Fig. 5). Each module represents a particular sub-model of knowledge. The main modules are the object-level domain knowledge model (real world entities and relationships), a strategy level model (for example a model of diagnostic problem solving), and two reasoning meta-level models, one for combining case-based and other types of reasoning, and one for combined learning methods. CREEK integrates problem solving and learning into one functional architecture.

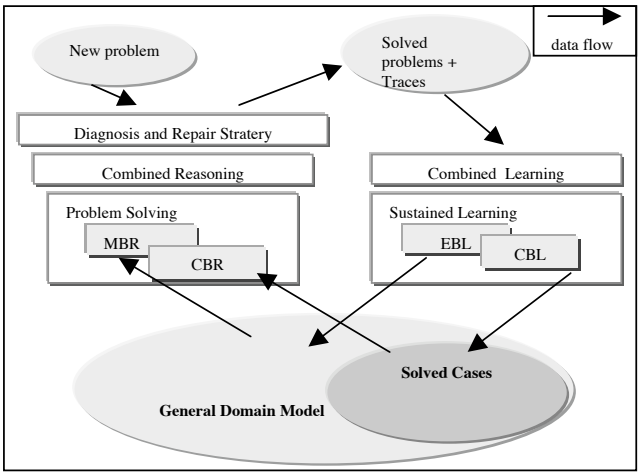


Fig. 5. The CREEK functional Architecture

Situation-specific experiences are held in the case base of solved cases. All the concepts are 'glued together' into a single, interconnected knowledge model. Diagnosis task concepts, for example, such as "symptom" and "diagnostic-hypothesis" (part of the diagnosis and repair strategy model), and learning task concepts, such as "case-indexing" and "failure-generalization" (part of the combined learning model), are defined within the same representation structure as general domain concepts like "appendicitis" and "fever", and case-related domains terms as "Patient#123456" and "current-radiation-dosage".

A knowledge model represented in CREEK is viewed as a semantic network, where each node and each link in the network is explicitly defined in its own frame. Each node in the network corresponds to a concept in the knowledge model, and each link corresponds to a relation between concepts. A concept may be a general definitional or prototypical concept, a case, or a heuristic rule, and describe knowledge of domain objects as well as problem solving methods and strategies. A frame represents a node in the network, i.e. a concept in the knowledge model. Each

concept is defined by its relations to other concepts, represented by the list of slots in the concept's frame definition. Fig. 6 illustrates the three main types of knowledge in CREEK, a top-level ontology of generic, domain-independent concepts, the general domain knowledge, and the set of cases.

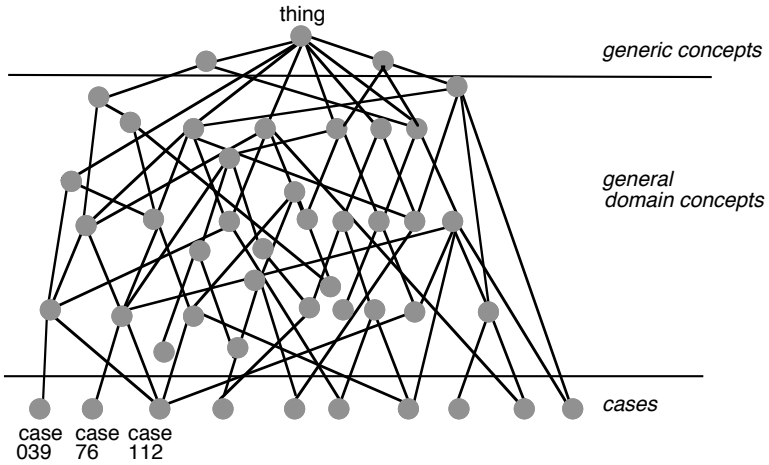


Fig. 6: Integrating cases and general knowledge

The case-based interpreter in CREEK contains a three-step process of 1) activating relevant parts of the semantic network 2) generating and explaining derived information within the activated knowledge structure, and 3) focusing towards and selecting a conclusion that conforms with the goal. This *activate-explain-focus* cycle, referred to as an 'explanation engine' [3], is a general mechanism that has been specialized for each of the four CBR tasks described in section 4, although the Revise task is not a system's task in CREEK. (see Fig. 7).

Fig. 7. The CBR process and the explanation engine

Similarity assessment is divided between the Activate and Explain steps of Retrieve. Activate first determines a relevant broad context for the problem, by spreading activation from goal concepts to relevant findings. Spreading-relations include general taxonomic ones, causal relations, associational relations, and application-specific relations. Only cases with activation strength above a certain threshold will be considered for further matching. The activation strength is based on the number of matched relations and their relevance factor, according to the following formula [15]:

$$sim(C_{IN}, C_{RE}) = \frac{\sum_{i=1}^{n}\sum_{j=1}^{m} sim(f_i, f_j) * relevancefactor_{f_j}}{\sum_{j=1}^{m} relevancefactor_{f_j}}$$

C_{IN} and C_{RE} are the input and retrieved cases, n is the number of findings in C_{IN}, m is the number of findings in C_{RE}, f_i is the i^{th} finding in C_{IN}, f_j the j^{th} finding in C_{RE}, and $sim(f_1,f_2)$ is simply given as:

$$sim(f_1, f_2) = \begin{cases} 1 & if\ f_1 = f_2 \\ 0 & otherwise \end{cases}$$

The relevance factor is a number that combines the predictive strength (degree of sufficiency) and importance (degree of necessity) of a feature for a stored case. Following Activate, Explain will attempt to improve the match between the input case and the activated cases. Only unmatched findings need to be explained, since the strength of the directly matched findings cannot be increased. Different explanation paths are combined [16] into a matching strength for each activated case. The paths have convergence points, i.e. explanatory concpets - such as causal concepts, for which there exist an explanation path from both findings. Its strength is the product of the strength of each relation leading from the finding to the convergence point:

$$path\ strength(f, c) = \prod_{i=1}^{n} relation\ strength_i$$

Here, n is the number of relations. There may exist one or more parallel paths from each finding to each convergence point. The resulting strength is based on the general formula for adding contributions from n parallel elements, S_1... S_n, into a total score:

$$parallel\ strength(S_1, S_2, ..S_n) = 1 - \prod_{i=1}^{n} (1 - S_i)$$

Thus, the total combined strength of all the paths leading from a finding *f* to a convergence point *c,* with *n* being the number of paths between *f* and *c,* is computed according to the following formula:

$$total\ path\ strength(f, c) = 1 - \prod_{i=1}^{n} (1 - path\ strength(f, c)_i)$$

The strength of one explanation path (eps) leading from a finding f1 to a finding f2 via the convergence point c, is computed by multiplying the total path strength for each of the findings to the convergence point, and the total explanation strength for the two findings (f1 and f2) via several convergence points is finally computed by using the parallel strength formula:

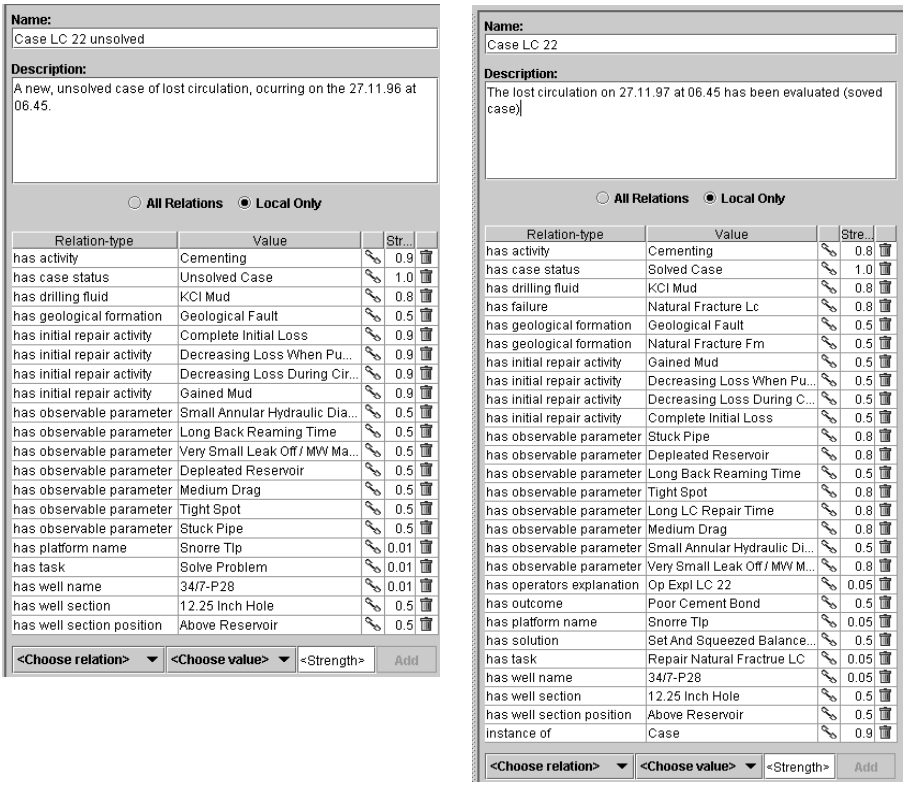
$$eps(f_1, f_2, c) = total\ path\ strength(f_1, c) \cdot total\ path\ strength(f_2, c)$$

$$\exp lanation\ strength(f_1, f_2) = 1 - \prod_{i=1}^{n} (1 - eps(f_1, f_2, c_i))$$

Here n is the number of convergence points between the findings, and c_i is the i^{th} convergence point.

Focus selects the best case or rejects all of them, based on the explanatory justification. It may adjust the ranking of the cases based on preferences or external constraints. The explanatory power of the domain model is also utilized in Reuse and Retain.

This was implemented in the former Lisp version. Research related to the current Java version – called TrollCreek - has focused on Retrieve, with Reuse in an early stage.



Name:
Case LC 22 unsolved

Description:
A new, unsolved case of lost circulation, ocurring on the 27.11.96 at 06.45.

All Relations | Local Only

Relation-type	Value	Str...
has activity	Cementing	0.9
has case status	Unsolved Case	1.0
has drilling fluid	KCl Mud	0.8
has geological formation	Geological Fault	0.5
has initial repair activity	Complete Initial Loss	0.9
has initial repair activity	Decreasing Loss When Pu...	0.9
has initial repair activity	Decreasing Loss During Cir...	0.9
has initial repair activity	Gained Mud	0.9
has observable parameter	Small Annular Hydraulic Dia...	0.5
has observable parameter	Long Back Reaming Time	0.5
has observable parameter	Very Small Leak Off / MW Ma...	0.5
has observable parameter	Depleated Reservoir	0.5
has observable parameter	Medium Drag	0.5
has observable parameter	Tight Spot	0.5
has observable parameter	Stuck Pipe	0.5
has platform name	Snorre Tlp	0.01
has task	Solve Problem	0.01
has well name	34/7-P28	0.01
has well section	12.25 Inch Hole	0.5
has well section position	Above Reservoir	0.5

<Choose relation> | <Choose value> | <Strength> | Add

Name:
Case LC 22

Description:
The lost circulation on 27.11.97 at 06.45 has been evaluated (soved case)

All Relations | Local Only

Relation-type	Value	Stre...
has activity	Cementing	0.8
has case status	Solved Case	1.0
has drilling fluid	KCl Mud	0.8
has failure	Natural Fracture Lc	0.8
has geological formation	Geological Fault	0.5
has geological formation	Natural Fracture Fm	0.5
has initial repair activity	Gained Mud	0.5
has initial repair activity	Decreasing Loss When Pu...	0.5
has initial repair activity	Decreasing Loss During C...	0.5
has initial repair activity	Complete Initial Loss	0.5
has observable parameter	Stuck Pipe	0.8
has observable parameter	Depleated Reservoir	0.8
has observable parameter	Long Back Reaming Time	0.5
has observable parameter	Tight Spot	0.8
has observable parameter	Long LC Repair Time	0.8
has observable parameter	Medium Drag	0.8
has observable parameter	Small Annular Hydraulic Di...	0.5
has observable parameter	Very Small Leak Off / MW M...	0.8
has operators explanation	Op Expl LC 22	0.05
has outcome	Poor Cement Bond	0.5
has platform name	Snorre Tlp	0.05
has solution	Set And Squeezed Balance...	0.5
has task	Repair Natural Fractrue LC	0.05
has well name	34/7-P28	0.05
has well section	12.25 Inch Hole	0.5
has well section position	Above Reservoir	0.5
instance of	Case	0.9

<Choose relation> | <Choose value> | <Strength> | Add

Fig. 8. Unsolved case (left) and the corresponding solved case (right) of Case LC 22.

The general domain knowledge is assumed to be extensive enough to provide sufficient support to the case-based methods, but may also provide a back-up capability of problem solving on its own, if no similar case is found. The general domain knowledge is typically built up by rather 'deep' relationships - for example a combination of causal, structural, and functional relations. It contains a simple model-based casual reasoning method, in addition to the basic inference methods of frame matching, constraint propagation, and plausible inheritance (see next chapter).

The TrollCreek tool allows running the case matching process at any time during system development. To illustrate, assume that we are on an oil rig in the North Sea. Drilling fluid losses have been observed, and the situation turns into a problem (so-called Lost Circulation). See the case description to the left in Fig. 8. TrollCreek produces first of all a list of similar cases for review of the user. Testing of Case LC 22 suggests that Case LC 40 is the best match, with case 25 as the second best, and with a matching degree of 45% - as shown in Fig. 9. Examination of these cases reveals that Case LC 40 and 25 are both of the failure type Natural Fracture (an uncommon failure in our case base). By studying Case LC 40 and 25 the optimal

treatment of the new problem is devised, and a new case (Case LC 22) is stored in the case base (right part, Fig. 8).

The user can choose to accept the delivered results, or construct a solution by combining several matched cases. The user may also trigger a new matching process, after having added (or deleted) information in the problem case. The user can also browse the case base, for example by asking for cases containing one specific or a combination of attributes. Figure 4 shows parts of the explanation of why Case LC 22 is a problem of the type Natural Fracture. The interactive graph displays the part of the semantic network that was involved in the matching, either by direct or indirect (explained) matches. A textual explanation of an indirect match is also displayed, as shown to the middle right in Fig. 9.

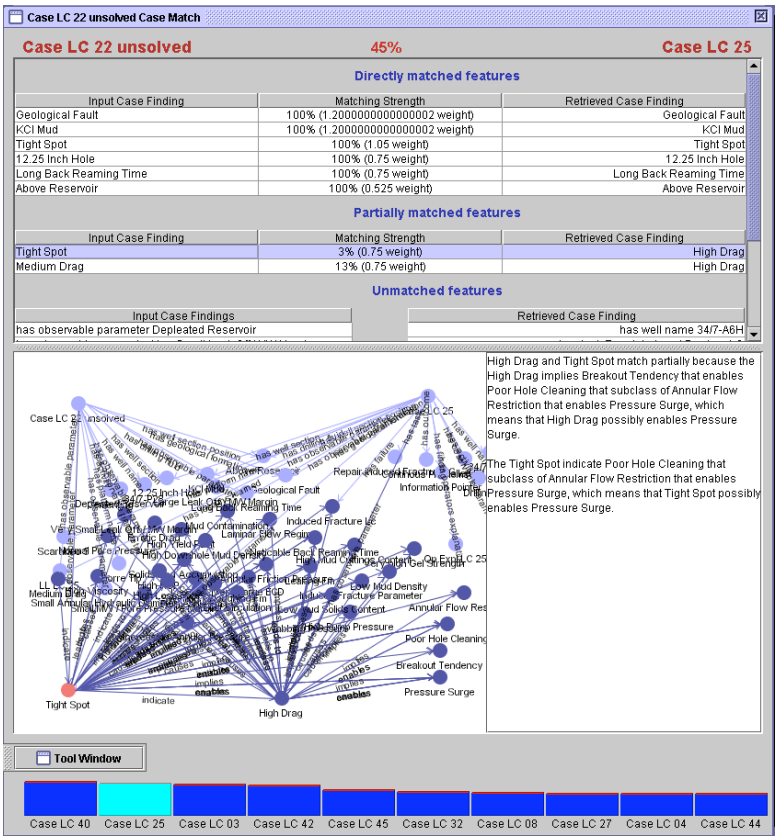


Fig. 9. Results of matching a new case (Case LC 22 unsolved) with the case base.

6. Recent and ongoing research

The transition to the Java platform, from Lisp, led us to make a revision of the

knowledge representation and basic inference methods. An earlier idea of plausible inheritance as an inference method for semantic networks [14] was generalized and made into the core model-based inference method of CREEK [16]. The main principle is that inheritance is extended to be applicable to any pair of a relationship and a relation, as opposed to inheritance only along a subclass relation. A location relationship may be inherited along a part-of relation, for example – assuming that parts of things are in the same location as the thing itself. Fig 10. illustrates how an initial frame, "epidemic case #3", having a local subclass relationship with "bacterial epidemic", and a causal relationship with "dirty water", inherits additional relationships (the R_i set at the lower right), through as set of inheritance rules (the I set at the lower left).

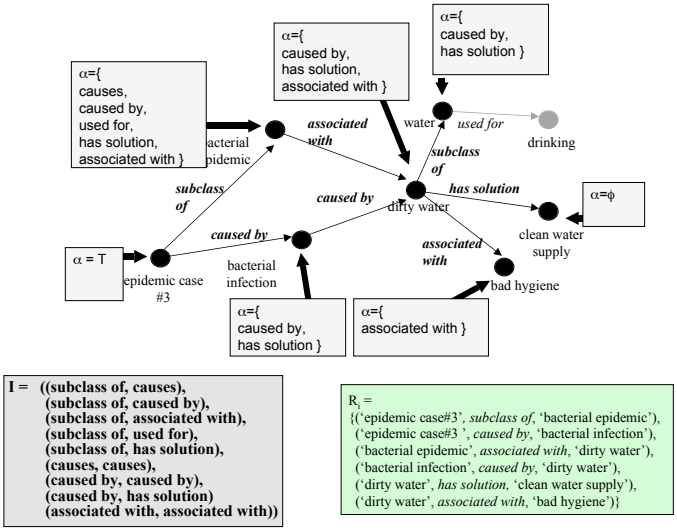


Fig. 10. Plausible inheritance example

Some current activities explore this method by designing systems where model-based reasoning play a strong part in itself, rather than only as part of the CBR process. Examples are two PhD projects where one is a method for generating and evaluating explanations for intelligent tutoring [17], and the other a method for generating explanations for gene–gene relationships and dependencies in order to understand the development of diseases at the level of functional genomics [18]. Our research into knowledge-intensive case-based explanation, studies the combined use of case-specific and general domain knowledge from the perspective of user-targeted explanations (the two projects just mentioned), as well from the perspective of the system-internal explanation methods in CREEK. The transparency of the knowledge representation system in CREEK favours studies of mutual explanation mechanism, i.e. explanation methods serving both purposes. This is currently studied within a PhD project on conversational case-based reasoning for software component reuse [19], although the focus here is on internal explanations within a CCBR context. Quite another issue is studied in a PhD research done within the EU project Ambiesense

[20], where an agent-based architecture is developed for CREEK, aimed to provide contextualized information to mobile users on business or tourist travels. Agent-based methods are also explored by others, which should lead to a generic distributed architecture for CREEK. Cases are used to personalize the information provided. A thorough study of context modeling was done in an earlier research applied to the medical diagnosis area [21]. This work, as well as a study done in medical image understanding [22], also made significant contributions to the knowledge-level modeling approach within CREEK.

Additional methods for representing and reasoning with general domain knowledge are also being explored. In particular, the studies of Bayesian Networks within CREEK [23] has given additional insights to the knowledge modeling and representation issues, as well as triggered studies on data mining methods for learning of general domain knowledge. Examples of smaller project that have developed additional demonstrators as part of MSc works, include an ANN system integrated into CREEK [24], for face recognition, and a text mining system for extracting general domain relationships from text [25]. Sometimes, it is also useful to lean back and take a look at the more fundamental issues related to developing CBR systems and other AI systems, such as relating current practice to totally different development and modeling views, such as one suggested by an autopoietic analysis [26].

7. Conclusion

The paper has described the knowledge-intensive CBR approach that is at the core of the CREEK framework, architecture, and system. By starting out with the fundamental issues related to the nature of knowledge, and the modeling-perspective taken to the development of a CREEK knowledge base, the actual representation and reasoning methods – as exemplified by Retrieve – hopefully become clearer.

The current and future directions of research focus more strongly on experimental evaluation of the various methods of CREEK. Of special interest currently, are experimentations related to the combined explanatory power of general domain knowledge and cases. This includes more thorough studies of the representation and basic inferencing methods. In addition, multi-agent architectures, text mining of general and case-specific knowledge, and conversational CBR methods, are high up on the research agenda. Finally, continued tool development, in connection with development of real world applications, is a priority, for which we cooperate with the company Trollhetta AS.

References

1. Díaz-Agudo, B., González-Calero, P.A.: An Architecture for Knowledge Intensive CBR Systems. In Blanzieri, E., Portinale, L., (Eds.): Advances in Case-Based Reasoning (Procs. of the 5th European Workshop on Case-Based Reasoning, EWCBR 2000), Lecture Notes in Artificial Intelligence, 1898, Springer, 2000.

2. Aamodt A.: A Knowledge-Intensive Integrated Approach to Problem Solving and Sustained Learning. PhD. Dissertation. University of Trondheim, Department of Electrical Engineering and Computer Science, Trondheim (1991). [Downloadable from authors publications homepage].
3. Aamodt, A. 1994: Explanation-driven case-based reasoning. In *Topics in case-based reasoning*, edited by S. Wess et al., Springer Verlag, 274-288.
4. Jære, M.D., Aamodt, A., Skalle, P.: Representing temporal knowledge for case-based prediction. *Advances in case-based reasoning; 6th European Conference, ECCBR 2002*, Aberdeen, September 2002. Lecture Notes in Artificial Intelligence, LNAI 2416, Springer, pp. 174-188.
5. Skalle, P., Aamodt, A.: Knowledge-based decision support in oil well drilling; Combining general and case-specific knowledge for problem solving. To appear in *Proceedings of ICIIP-2004*, International Conference on Intelligent Information Processing, Beijing, October 2004.
6. Clancey, W.J.: The frame of reference problem in the design of intelligent machines, In K. VanLehn (ed.), *Architectures for Intelligence.* Lawrence Erlbaum, 1991, p. 357-423.
7. Agnar Aamodt, Mads Nygaard: Different roles and mutual dependencies of data, information, and knowledge - an AI perspective on their integration, *Data and Knowledge Enigneering* 16 (1995), pp 191-222
8. C.G. Hempel, *Aspects of scientific explanation.*, (Free Press, New York, 1965).
9. P. Thagard, *Computational Philosophy of Science*, (MIT Press/Bradford Books, 1988).
10. Clancey W.J.: Viewing knowldge bases as qualitative models. *IEEE Expert*, Vol.4, no.2. Summer 1989. pp. 9-23.
11. Newell, A.: The knowledge level, *Artificial Intelligence*, 18 (1982) 87-127.
12. Van de Velde, W.: Issues in knowledge level modelling, In J-M. David, J-P. Krivine, R. Simmons (eds.), *Second generation expert systems* (Spinger, 1993) 211-231.
13. Aamodt, A.: Modeling the knowledge contents of CBR systems. *Proceedings of the Workshop Program at the Fourth International Conference on Case-Based Reasoning*, Vancouver, 2001. Naval Research Laboratory Technical Note AIC-01-003, pp. 32-37.
14. Aamodt A.: A Knowledge Representation System for Integration of General and Case-Specific Knowledge. Proceedings from IEEE *TAI-94, International Conference on Tools with Artificial Intelligence* (1994). New Orleans, November 5-12.
15. Lippe, E.: Learning support by reasoning with structured cases. MSc Thesis, Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, 2001.
16. Sørmo F.: Plausible Inheritance; Semantic Network Inference for Case-Based Reasoning. MSc thesis, Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, 2000.
17. Frode Sørmo, Agnar Aamodt: Knowledge communication and CBR. 6th European Conference on Case-Based Reasoning, ECCBR 2002, Aberdeen, September 2002. Workshop proceedings. Robert Gordon University, pp. 47-59.
18. Waclaw Kusnierczyk, Agnar Aamodt and Astrid Lægreid: Towards Automated Explanation of Gene-Gene Relationships. RECOMB 2004, The Eighth International Conference on Computational Molecular Biology, Poster Presentations, E9, San Diego, March 2004.
19. Gu, M., Aamodt, A., Tong, X.: Component retrieval using conversational case-based reasoning. To appear in *Proceedings of ICIIP-2004*, International Conference on Intelligent Information Processing, Beijing, October 2004.
20. Anders Kofod-Petersen, Agnar Aamodt: Case-based situation assessment in a mobile context-aware system. Proceedings of AIMS2003, Workshop on Artificial Intgelligence for Mobil Systems, Seattle, October, 2003.

21. Pinar Ozturk, Agnar Aamodt: A context model for knowledge-intensive case-based reasoning. International Journal of Human Computer Studies. Vol. 48, 1998. Academic Press. pp 331-355.
22. Morten Grimnes, Agnar Aamodt: A two layercase-based reasoning architecture for medical image understanding. In Smith, I., Faltings, B. (eds). *Advances in case-based reasoning, (Proc. EWCBR-96),* Springer Verlag, Lecture Notes in Artificial Intelligence 1168, 1996. pp 164-178.
23. Helge Langseth, Agnar Aamodt, Ole Martin Winnem: Learning retrieval knowledge from data. In Sixteenth International Joint Conference on Artificial Intelligence, Workshop ML-5: Automating the Construction of Case-Based Reasoners. Stockholm 1999. Sarabjot Singh Anand, Agnar Aamodt, David W. Aha (eds.). pp. 77-82.
24. Engelsli, S.E.: Intergration of Neural Networks in Knowledge - Intensive CBR. MSc thesis, Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, 2003.
25. Tomassen, S.L.: Semi-automatic generation of ontologies for knowledge-intensive CBR. MSc thesis, Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, 2003.
26. Sverberg, P.: Steps towards an empirically responsible AI; A theoretical and methodological framework. MSc thesis, Norwegian University of Science and Technology (NTNU), Department of Computer and Information Science, 2004.