

# Knowledge-Intensive Case-Based Reasoning and Sustained Learning

Agnar Aamodt

Knowledge Engineering Laboratory, ELAB-RUNIT, SINTEF  
N-7034 Trondheim-NTH, Norway  
and

Department of Electrical Engineering and Computer Science  
University of Trondheim.  
agnar.aamodt@elab-runit.sintef.no

## Abstract

In case-based reasoning (CBR) a problem is solved by matching the problem description to a previously solved case, using the past solution in solving the new problem. A case-based reasoner learns after each problem solving session by retaining relevant information from a problem just solved, making the new experience available for future problem solving. Crucial steps in a CBR process include finding a good match to a new problem, adapting a previous solution to successfully solve the new problem, and deciding how to index and store a new case for later effective retrieval. Previous CBR systems relied on syntactic rather than semantic or pragmatic criteria in performing these steps. A comprehensive model of general domain knowledge is needed in order to match cases based on their meaning contents. This has led to systems that attempt to combined case-based methods with model-based - explanation-based - approaches. Two systems representative of this research, PROTOS and CASEY, are briefly described. The systems are discussed with respect to what type of general knowledge they contain, and the degree of model-based support for the case-based reasoning and learning processes they exhibit. Some of their weaknesses are identified, and an improved approach to integration of case-based and model-based methods - called CREEK<sup>1</sup> - is suggested.

## 1. Introduction

In order to meet the challenge of problem solving in real-world, open domains, future knowledge-based systems need to become more *competent* and *robust*. These systems will operate within a continually evolving environment, and their knowledge therefore needs continuous updating and refinement. Methods for automated *learning from experience* are strongly wanted. The ability to learn from each problem solving experience is here referred to as *sustained learning*.

Problem solving in the real world involves *open problems*. An open problem is characterized by having a weak or intractable domain theory. Typical weak theory domains are medical diagnosis, geological interpretation, investment planning, and most engineering domains. A weak domain theory is characterized by *uncertain relationships* between its concepts. The stronger the theory, the more certain are its relationships. At the very far end of this spectrum are the *complete theories*, with only certain relationships, e.g. as expressed by standard logical relationships based on truth and falsity.

When presented with a new problem, a person is often reminded

of a previous problem similar to the one at hand. For example: A physician - after having examined a particular patient in his office - gets a reminding to a patient that he treated two weeks ago. If the reminding was caused by a similarity of important symptoms (and not the color of the patient's sweater, say), the physician uses the diagnosis and treatment of the previous patient to determine the disease and cure for the patient in front of him. A financial consultant working on a difficult credit decision task, uses a reminding to a previous case, which involved a company in similar trouble as the current one, to recommend that the loan application should be refused. A drilling engineer, who have experienced two dramatic blow out situations, is quickly reminded of one of these situations (or both) when the combination of critical measurements matches those of a blow out case.

The examples above illustrate that reasoning by re-using or modifying past experiences is a powerful and frequently applied paradigm for human problem solving. This claim is also supported by results from cognitive psychological research. Schank [Schank-82] developed a theory of learning and reminding based on retaining of experience in a dynamic, evolving memory structure. Anderson [Anderson-83] has shown that people use past cases as models when learning to solve problems, particularly in early learning. Other results (e.g. by W.B. Rouse, as described in [Kolodner-85]) indicate that the use of past cases is a predominant problem solving method among experts as well<sup>2</sup>. Studies of problem solving by analogy (e.g. [Gentner-83, Carbonell-83]) also shows the frequent use of past experience in solving new and different problems. Analogy reasoning is closely related to case-based reasoning, but the study issues focused are different in the two disciplines: While a main research issue in analogy [Hall-89] is the mapping of a new problem description to a known problem in a different domain, case-based methods focus on indexing and matching strategies for single-domain cases [Burstein-89].

The driving force behind case-based methods in AI comes from the machine learning community, and case-based reasoning (CBR) is regarded a subfield of machine learning [Rissland-89]. In CBR learning occurs as a natural 'by-product' of problem solving.

---

<sup>2</sup>Anderson's domain was geometry problems, a well defined domain with a complete domain theory, while Rouse observed trouble-shooting by steam plant operators, a more open domain with an incomplete domain theory. Different characteristic of the domains may explain the different results of these experiments.

---

<sup>1</sup>Case-based Reasoning through Extensive Explicit Knowledge.

The CBR paradigm covers a range of different methods for organizing, indexing, retrieving and utilizing the knowledge retained in past cases. Cases may be kept as separate instances, or a set of similar cases may form a generalized case. Cases may be indexed by a pre-fixed or an open vocabulary, and within a flat or hierarchical index structure. The solution from a past case may be directly applied to the present problem, or modified according to differences between the two cases.

While early case-based approaches retrieved previous cases based on superficial, syntactic similarities among problem descriptors (e.g. the CYRUS system, described in [Kolodner-83a] and [Kolodner-83b]), some recent approaches attempt to perform matching based on *relevant features* that are *semantically similar* (as in the PROTOS [Bareiss-88], CASEY [Koton-89] and GREBE [Branting-89] systems). In order to match cases based on semantic similarities and relative importance of features, an extensive body of generalized domain knowledge is needed to produce a sufficient *explanation* of why two cases match, and how strong the match is.

## 2. System requirements

Our claim is that future knowledge-based systems - intended to solve real-world open problems and to continually learn from experience - should fulfil the following three requirements:

1. An expressive and extendible knowledge representation formalism, enabling an explicit and thorough modelling of all relevant knowledge types.
2. A problem solving and reasoning method that is able to effectively combine reasoning from past cases with reasoning within a competent and robust model of more general domain knowledge.
3. A learning method that is able to retain concrete problem solving experiences, and integrate each experience into the knowledge model in a way that gradually improves knowledge quality and problem solving performance.

## 3. Current systems

We have performed a study of existing knowledge-intensive case-based reasoning systems, and analyzed them with respect to these requirements. The two systems that were considered to fit the specifications most closely are reviewed in this chapter, followed by a suggested improved approach - the Creek architecture<sup>1</sup>. All three systems will be described in four parts: General overview, knowledge representation, reasoning, and learning method.

### 3.1. PROTOS

Protos [Bareiss-88, Porter-86] is a case-based<sup>2</sup> approach to

---

<sup>1</sup>The analysis, and the development of CREEK, is part of a Ph.D research now in its terminal phase, supervised by Arne Sølvberg. Other systems that have been analyzed as part of this research are the CHEF [Hammond-86] and JULIA [Kolodner-87] systems.

<sup>2</sup>Actually, Protos is called an *exemplar-based* system, emphasizing that all cases are stored exclusively as concrete, non-generalised exemplars. A concept

concept learning and classification problem solving. A system for diagnosing hearing disorders has been developed.

A problem presented to Protos is described as a set of features, and the system's task is to retrieve the previous case that best matches the feature set. Cases are indexed by *reminders* from features. The category of a retrieved case is proposed as a solution to the problem, without adaption. If the proposed solution is rejected by the user, a learning session is initiated. Protos may be asked to look for another solution or to accept a solution from the user. The user is forced to define entered terms that are unknown to Protos, by describing their relations with existing terms. In this way general domain knowledge is gradually built up.

General domain knowledge is represented as a semantic network of categories, features and relations. A category is represented by its set of member cases, and its set of links to other categories.

An case is represented by its set of features, and the category to which it belongs. Each feature in a stored case is associated with a numerical importance value. This value expresses how important the feature is for classifying the case as a member of the category. For example, the feature 'backrest' is more important than 'wheels' for the category 'chairs'.

A comprehensive set of relations are defined (e.g. 'part-of', 'specialization-of', 'causes', 'enables', 'suggests') where each relation has a number of explanatory strength associated. An explanation is a chain of relationships between two features or between a feature and a category. An explanation is accepted if its strength - calculated by combining strengths of each relation in the chain - is above some threshold value.

Compared to the requirements, the representational system has two major weaknesses: First, features and categories are not structured concepts composed of properties with values, but flat property-value pairs or single values. Second, relations are not regarded as concepts, i.e. as knowledge to be reasoned about and learned. The set of relations can not be extended, their pre-defined explanation strengths not modified, etc., without changing the Protos program code.

The reasoning method is case-based, where the sub-processes of feature matching, and of evaluating a feature's relevance to a category, are supported by semantic network model reasoning. There are no methods that use the knowledge model to derive a solution if the case-based method fails. Instead, the user is asked to solve the problem. Neither is general domain knowledge used to justify that a solution is applicable to the new problem, this is also left to the user.

Protos always learn from a problem solving case: If a problem is successfully solved in the first attempt, no new case is constructed, but the reminders from relevant features to the case is strengthened. If a problem is successfully solved in second or later attempts, Protos tries to find the cause of the initial failure. Protos learns from the failure by weakening reminders from the features to the faulty retrieved case. If Protos is unable to suggest a solution, the case is stored as a new case. Reminders to the case, and difference links to similar

---

definition is viewed extensionally, as a category, defined by the collection of cases (exemplars) that belong to the category.

cases, are installed. During the learning process, the user is asked to confirm or change suggested modifications to the case structure, and to revise explanations if needed.

Protos is a learning apprentice that relies heavily on its user. This is both a strength and a weakness. A positive effect is a quick adaption to the real-world problem environment; the system will always be up to date with knowledge related to cases it has recently seen. The major weakness is that the knowledge model of the system eventually will represent a resource that is only partially utilized.

## 3.2. CASEY

CASEY [Koton-88, Koton-89] is a system that combines case-based and model-based reasoning. When a problem turns out to be unsolvable by retrieving a past case, a general domain knowledge model is used in a second attempt to solve the problem. The domain model also plays an active part in supporting the case-based reasoning and learning processes. Type of problems addressed are diagnosis of heart diseases. The general knowledge model in CASEY is a pure *causal model*., relating features to their causal states. CASEY's case memory structure is based on Kolodner's self-organizing memory system [Kolodner-83a].

A problem is solved by retrieving a case, and - as unlike Protos - *adapting* the past solution to the new problem. Each case contains a causal explanation that relates its features to the diagnosis. The solution to a new problem is derived by using the knowledge model to modify the explanation of the retrieved case.

Cases are stored in a dynamic memory structure as described in [Schank-82] and [Kolodner-83a]. The structure is a discrimination network, where the top node contains common properties of all cases in the structure. Downwards in the memory structure cases are indexed according to their differences with other cases. The cases themselves are leaves in the tangled tree-structure. An intermediate node represents a generalised description of the cases indexed under the particular node. A feature is regarded more general than another if it is contained within more cases than the other.

Concerning expressiveness, the only relation for deep modelling is 'causes'. The only moderator of the causal relation is a numeric probability - or likelihood - measure. This measure does not capture the underlying reasons for one cause being more plausible than another. Expressiveness is also limited by features and states being flat (property-name property-value) pairs, with no structuring of properties.

The reasoning method is a combination of case-based and model-based reasoning. The case based method is applied first, model-based reasoning within the causal model is performed if the case method fails to find a sufficiently similar past case. In addition to being a separate reasoning method, model-based reasoning also supports the case-based process.

A weakness of reasoning and problem solving is the lack of interaction with the user. A real-world problem solver should be able to come up with expectations and other consequences which it may want to check with the user.

CASEY always learns from a problem solving case: If a problem is successfully solved by case-based reasoning, CASEY stores the new case if it has significant features different from the previous case. If the new case is identical to the previous one, information about the importance of a feature is updated. If case-based reasoning fails, and the causal model solves the problem, a new case is created and indexed in the memory structure. The learning in CASEY does not involve user interaction. The system is designed to improve performance efficiency of model-based reasoning within the deep heart failure model. CASEY learns associational, compiled knowledge by extending or modifying<sup>1</sup> its case base.

A major weakness of the learning process is the 'closed world' within which CASEY operates. Only cases are learned, since the general domain knowledge is regarded as fixed. A second weakness is the limitation on knowledge intensive learning imposed by the restricted domain model. For example, CASEY attempts to generalise a causal feature when learning an explanation, but the only generalisation possible is along causal links.

## 4. CREEK

### General Architecture

The Creek architecture<sup>2</sup> is based on an integrated computational model of problem solving and learning. The architecture contains four modules, integrated within a common conceptual fundament. Each module represents a particular sub-model of knowledge: An object-level *domain knowledge model*, and three meta-level knowledge models: A *model of diagnosis and repair*, a *model of combined reasoning* (combining case-based, model-based and rule-based reasoning), and a *model of sustained learning*. The common underlying model - called the *conceptual knowledge fundament* - defines all concepts (i.e. all entities and relations) within a single, common knowledge model. Thus, diagnosis task concepts (e.g. symptom, diagnostic-hypothesis) as well as learning task concepts (e.g. case-index, failure-generalization), are defined within the same representational structure as domain concepts (e.g. car, weak-battery, has-color). The conceptual knowledge fundament glues the modules together by being a common pool of concept definitions for all the terms used in the four models. Each of the four sub-models contain operational knowledge like heuristic associations (cases and rules), and procedures (e.g. strategies, action sequences). The object level domain model contains the collection of past cases, but may also include a manually defined and maintained set of heuristic rules.

The diagnosis and repair model breaks the problem solving process into a set of tasks (e.g. get-test-result, select-relevant-symptom, generate-hypothesis, infer-expectations), and a model of static and dynamic interdependences between the tasks.

The combined reasoning model decides what type of reasoning - and associated domain knowledge - to use, given a particular state of the system. The kind of concepts defined at this level are concept-network, rule, case, model-based-reasoning, case-based-

---

<sup>1</sup>Modification of feature importances, measured by the number of times a feature is seen in case, and the number of times it is used in a causal explanation.

<sup>2</sup>Background, motivation., and a more general overview of Creek is given in [Aamodt-89a] and [Aamodt-89b]. The underlying computational model is described in [Aamodt-90].

reasoning.

The sustained learning model basically contains the algorithms for knowledge-supported case based learning, and a collection of rules to guide the matching and generalisation processes.

Creek integrates problem solving and learning into one functional architecture. Problem solving in Creek is performed by a combination of model-based, case-based and rule-based reasoning. The learning combines case-based and explanation-based methods. The flow of control and information between the knowledge base and the processes of problem solving and learning is shown in the figure below.

### Knowledge representation

All types of knowledge in Creek are represented in frames<sup>1</sup>. An important characteristic is that relations as well as entities are regarded as concepts, explicitly defined in their own frames. Creek has an internal model of its representational structure, i.e. a meta-level model containing concepts such as 'case-frame', 'rule-frame', 'slot', 'transitive-slot', 'constraint-facet', etc.

A general concept in Creek is described by its *typical properties*, which are inherited by more specialized concepts and instances. The inheritance is not absolute; an inherited property may be overridden by a specific, local property. The basic inference method is, thus, *default inheritance*. In addition, more restricted inheritance methods, such as forced inheritance and condition dependent inheritance, are also available. A more complex inference method is *frame matching*. While property inheritance is a method for inferring a value given a particular concept, frame matching infers ('retrieves') a concept given one or more property values. A third inference method of importance to frame systems is *constraint enforcement*. Constraints are specified as facets and used to check possible values, and to perform constraint-based reasoning steps. Some types of constraints - e.g. a value class - may be used to infer the superclass of a value if no specific value is inferable. Constraints may be propagated along specified relations, and inherited and combined into more specific constraints, from which conclusions may be inferred, explanations generated, and focused questions presented to the user.

It is important to note that the kind of frame system paradigm adopted in Creek reflects a non-classical view to concept definition. While a classical definition defines a concept as a set of necessary and sufficient properties, the frame system in Creek defines a concept in terms of a prototype - i.e. as a set of typical properties. Thus, our notion of a frame system is different from frame representations based on predicate logic, like the KL-ONE [Brachman-79] and KRYPTON [Brachman-85] systems, which do not have default inheritance.

The memory structure of past cases has strong similarities with the Protos model. The differences are due to the more expressive representation language of Creek, and the combined reasoning approach, which enables effective utilization of additional case information. In addition to input features and the suggested solution, a Creek case contains, e.g., the solution's explanation

path, whether a suggested diagnosis was successful or not, and why a solution was unsuccessful<sup>2</sup>.

### Reasoning

The reasoning control component (see the figure) is the operational part of the combined reasoning model. One of its tasks is to decide whether the case base or the rules should be used in the initial attempt to solve the problem by associational (non-deep) methods. When presented with a new problem, the reasoning controller receives a problem frame containing the input features and the current goal. In a system for diagnosis of car troubles, for example, the initial goal could be find-starting-fault. If a feature is known to the system, the feature will have a frame that describes its current properties (e.g. its parent classes, constraints, relations, associated faults, cases reminded of). If an unknown feature is presented, the system asks for a description of the feature, and creates the necessary frames. The system checks whether this description violates any constraints, assumptions or current facts. This process should be viewed as an attempt to understand the problem by integrating its description into the knowledge base. The method used is similar to methods for *knowledge integration*, as described by [Murray-88] and [Eggen-90].

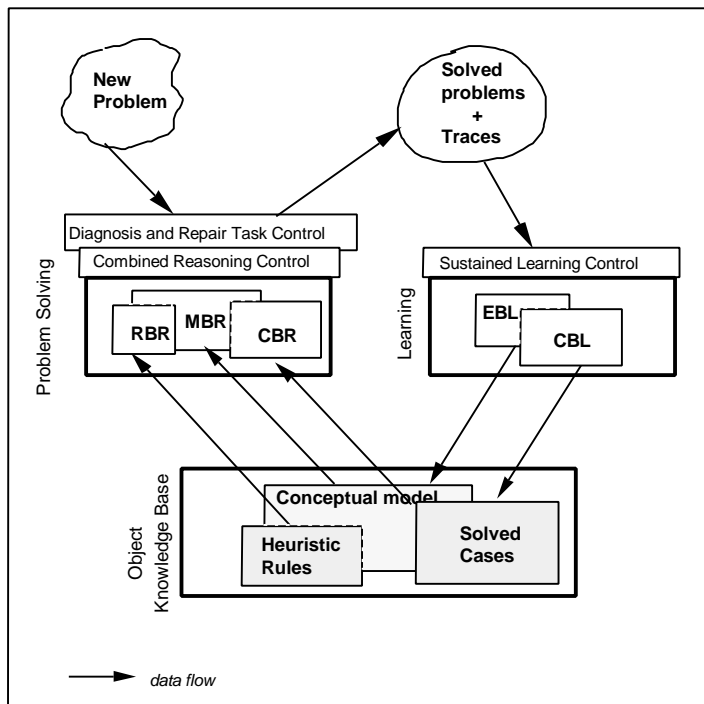
If the features (input or derived) give a reminding to a previous case that is above a particular strength level, case based problem solving is tried, otherwise the rule base is activated. The value of this *threshold level* depends on the relative contents and power of the case based and rule based subparts. The initial threshold value is set manually, and adjusted over time according to how well the chosen reasoning paradigm performs. For example, if an application has a well developed collection of rules, but has not seen many problems yet, the user may want to optimize performance by keeping the threshold level high. The rule-base will then be used in the initial attempt to solve the problem, unless there is a very strong reminding to a case. As more cases are added, the ratio of case-based to rule-based successes will increase, and the system will gradually lower the threshold value. Elaborate methods based solely on deeper model knowledge are tried only after the rules, too, have failed.

---

<sup>1</sup>The notion of a frame system - as introduced by Minsky [Minsky-75], and further developed by a number of authors [e.g., Bobrow-77, Greiner-80, Lenat-86] - also implies a set of characteristic inference methods that operate on the knowledge structure.

---

<sup>2</sup>A retrieved case may then be used to avoid the failure next time; the explanation of the failure serves to improve the matching of the two cases with respect to the failure.



**Figure: Creek's Functional Architecture**

At a high level of abstraction, Creek is an integrated architecture containing three building blocks: An object level knowledge base, a problem solver and a learner. The object knowledge base contains a conceptual knowledge model, and a collection of past cases. Heuristic rules are incorporated as part of the conceptual model. Creek contains multiple methods for reasoning and learning, where each method communicate with a part of the knowledge base.

The basic reasoning model in Creek is a three step process of *activate - explain - focus*. Upon reading a problem description, the system activates an initial knowledge structure by marking network concepts that matches terms of the input description as active. Other concepts are activated by a mechanism called *goal-focused spreading activation*<sup>1</sup>. This is a two step method of constraining the spreading of activation in a semantic network:

First, the set of concepts representing the goal of the reasoning process is activated. Let us name this set  $G_I$ . Activating the goal triggers activation of other concept nodes by following certain relational links. In this way, activation is spread to concepts in the neighbourhood of the goal concepts along a subset of the relations attached to goal concepts (e.g. causal, functional and structural relations). The set of concepts that have been activated when this cascade of activations terminates,  $G_A$ , are possible candidates for activation by spreading from the problem description concepts.

In the second step, the nodes represented by the problem descriptors are activated, forming the concept set  $P_I$ . The activation is propagated, along particular relations, out from the input descriptors. Any such path that hits a  $G_A$  node (making it a 'hit node') is assumed to contain nodes relevant to the current context, forming the node set  $P_A$ . The output from the activation

process is the union of  $P_A$  and the subset of  $G_A$  lying on paths between the 'hit nodes' and  $G_I$ .

The explanations for support of case-based reasoning - or for driving model-based reasoning - are produced within the activated part of the knowledge structure. A specialization of the activate-explain-focus cycle controls the case-based reasoning part, by activating a set of possibly matching cases, refining this set by explaining the match of each non-identical feature, and focusing on a solution by selecting the best matching case and copying or modifying the solution of the old case.

### Learning

The Learning Control component of the architecture (see the figure) is the operational part of the sustained learning model. The learning controller guides the learning process by providing a strategy for when to perform particular learning tasks, and how to combine tasks. Creek learns from every problem solving experience. If a successful solution was copied from a previous case, the reminding to that case from each relevant feature is strengthened. No new case is stored, but an attempt is made to merge the two cases by generalizing their feature-values. The specific features are not removed after generalization, but kept in a particular facet under the feature name, in case the generalized value later proves to be an over-generalization. For a generalized value to be acceptable, it has to be explained by the deep model, and it must not be a feature value in another case which is on the current case's difference list.

The main target for the learning process in Creek is the case base. But Creek may also learn general knowledge through interaction with the user during problem solving. There is no inductive learning of explicit, generalized concept definitions or rules in Creek<sup>2</sup>. The only type of generalization attempted before a case is stored is feature generalization. The rule-base is primarily used for associational, rule-based reasoning. However, since heuristic rules are represented within the conceptual model, they are available for the same tasks as the conceptual model in general. A rule is viewed as a shallow type of relation, and may be used to support learning, for example. Even if the explanatory strength of a shallow relation in general is low, it may add to other explanations for the same hypothesis and, thus, contribute to a justification of an hypothesis.

If a solution was derived by modifying a previous solution, a new case is stored and difference links between the two cases are established. A new case is also created after a problem has been solved from rules or from within the deeper knowledge model. Creek is, so far, basically an architecture - a system design framework. The application in focus during development of Creek has been diagnosis and treatment of oil drilling fluid (mud). Some basic components are implemented (in CommonLisp) - including the frame representation system<sup>3</sup> with basic inference and spreading activation methods. As a part of his thesis research on combined case-based and inductive learning Velitchko Koulitchev [Koulichev-90] wrote a case

<sup>2</sup>Generalization are *implicitly* learned, however, since explanation-based matching of cases implies a common generalization of the features - or feature sets - that match.

<sup>3</sup>The frame system in Creek is an extension of the METAKREK frame system [Solvberg-88, Vestli-89, Nordbø-89]. The system is linked to the graphical knowledge editing tool of METAKREK [Aakvik-88], enabling visualisation and interactive modification of knowledge structures on the screen (implemeted on a TI Explorer and ported to Sun 4).

<sup>1</sup>The notion of spreading activation was introduced in [Quillian-68]. It is a basic mechanism in Anderson's human cognitive architecture [Anderson-83] and in Thagard's computational model of scientific reasoning [Thagard-88].

matcher and generalizer, and Christian Larssen [Larssen-90] integrated a rule chainer within the frame representation system as part of his thesis research on multi-paradigm reasoning.

## 5. Conclusion

Compared to the specified requirements for competent and robust knowledge based systems that learn from experience, the Creek architecture represents a significant improvement over existing approaches: By focusing on knowledge maintenance, it puts a stronger emphasis on extensive use of the existing domain knowledge in its case-based reasoning and learning methods. It provides increased competence and robustness by offering multiple reasoning methods as well as problem solving 'from scratch' within the deeper, conceptual model. It is able to continually maintain its knowledge by learning from each problem case presented to it. The integration of multiple knowledge types and reasoning methods has led to incorporation of meta-level knowledge models that explicitly represent and reason with control-level concepts.

No studies has (yet) been undertaken regarding the performance efficiency of an application. The main goal of Creek has been to develop a knowledge-based system architecture that supports knowledge-intensive and case-based methods for reasoning and sustained learning. Highly expressive representation formalisms and complex reasoning schemes are, in general, computational expensive. There are, basically, three approaches to solving this problem: One is a computational approach, like investigating low level data access methods, parallel processing, etc. Another is a representational approach, i.e. to constrain the representation of knowledge and reasoning so it fits more readily to the underlying computational processes. The third approach emphasizes on the knowledge content, by trying to stabilize the size of a knowledge base by continually learning of generalized and operationalized - purpose directed - knowledge as more experience is gained.

## References

### Aakvik-88

Geir Aakvik, Mette Vestli, Inge Nordbø, Ingeborg Sølvsberg, Tore Amble, Agnar Aamodt: METAKREK; METATOOL documentation. ELAB-RUNIT Report STF14 A88055. SINTEF, Trondheim, 1988.

### Aamodt-89a

Agnar Aamodt: Towards robust expert systems that learn from experience - an architectural framework. In John Boose, Brian Gaines, Jean-Gabriel Ganascia (eds.): *EKAW-89; Third European Workshop on Knowledge-Based Systems*, Paris, July 1989. pp 311-326.

### Aamodt-89b

Agnar Aamodt: Towards expert systems that learn from experience. In: *Proceedings from the Case-Based Reasoning Workshop*, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann, 1989. pp 181-187.

### Aamodt-90

Agnar Aamodt: A computational model of knowledge-intensive problem solving and learning. In: *EKAW-90; Fourth European Knowledge Acquisition for Knowledge-Based Systems Workshop*, Amsterdam, June 25,29, 1990.

### Anderson-83

John R. Anderson: *The architecture of cognition*. Harvard University Press, Cambridge, 1983.

### Bareiss-88

Ray Bareiss: PROTON; a unified approach to concept representation, classification and learning. Ph.D Dissertation, University of Texas at Austin, Dep. of Comp. Sci. 1988. Technical Report AI88-83.

### Bobrow-77

Daniel Bobrow, Terry Winograd: An overview of KRL, a knowledge representation language. *Cognitive Science*, Vol. 1, no. 1, 1977. pp 3-46. (Also in R.J Brachman, H.J. Levesque: *Readings in knowledge representation*. Morgan Kaufmann, 1985. pp 263-285.)

### Brachman-85

Ronald Brachman, Richard Fikes, Hector Levesque: KRYPTON, A functional approach to knowledge representation. In: Ronald Brachman, Hector Levesque: *Readings in knowledge representation*. Morgan Kaufmann. 1985. pp 411-430.

### Brachman-79

Ronald Brachman: On the epistemological status of semantic networks. In N. V. Findler (ed.): *Associative networks; representation and use of knowledge by computers*. pp 3-50. Also in R. Brachman, H. Levesque: *Readings in knowledge representation*. Morgan Kaufmann. 1985. pp 191-215.

### Branting-89

Karl Branting: Representing and reusing explanations of legal precedents. In *Proceedings, International Conference on Artificial Intelligence and Law*, Vancouver, June 13-16, 1989.

### Burstein-89

Mark H. Burstein: Analogy vs. CBR; The purpose of mapping. *Proceedings from the Case-Based Reasoning Workshop*, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann, 1989. pp 133-136.

### Carbonell-83

Jaime Carbonell: Learning by analogy - formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell, T.M. Mitchell (eds.): *Machine Learning - An artificial Intelligence Approach*, Vol.1, 1983. Morgan Kaufmann. pp137-161.

### Eggen-90

Jorun Eggen, Astrid M. Lundteigen, Mette Mehus: Integration of knowledge from different knowledge acquisition tools. In *EKAW-90, Fourth European Knowledge Acquisition for Knowledge-Based Systems Workshop*, Amsterdam, June, 1990.

### Gentner-83

Dedre Gentner: Structure mapping - a theoretical framework for analogy. *Cognitive Science*, Vol.7. s.155-170. 1983.

### Greiner-80

Russel Greiner, Doug Lenat: A representation language language. *Proceedings AAAI-80*, Morgan Kaufmann, 1980. pp. 165-169.

### Hall-89

Rogers P. Hall: Computational approaches to analogical reasoning; A comparative analysis. *Artificial Intelligence*, Vol. 39, no. 1, 1989. pp 39-120.

### Hammond-86

Kristion J. Hammond: CHEF; a model of case-based planning. *Proceedings of AAAI-86*. Morgan Kaufmann, 1986. pp 267-271.

### Kolodner-83a

Janet Kolodner: Maintaining organization in a dynamic long-term memory. *Cognitive Science*, Vol.7, s.243-280. 1983.

### Kolodner-83b

Janet Kolodner: Reconstructive memory, a computer model. *Cognitive Science*, Vol.7, s.281-328. 1983.

**Kolodner-85:** Experiential processes in natural problem solving. Report, Georgia Institute of Technology, GIT-ICS-85/23, October 1985.

### Kolodner-87

Janet Kolodner: Extending problem solver capabilities through case-based inference. *Proc. 4th workshop on Machine Learning*, UC-Irvine, June 22-25 1987. pp 167-178.

**Koton-88**

Phyllis Koton: Reasoning about evidence in causal explanations. *Proceedings of AAAI-88*. pp 256-261.

**Koton-89**

Phyllis Koton: Using experience in learning and problem solving. Massachusetts Institute of Technology, Laboratory of Computer Science (Ph.D. diss, October 1988). MIT/LCS/TR-441. 1989.

**Koulichev-90**

Velitchko Koulichev: Generalization in case-based machine learning. RIK 90-3, ELAB-RUNIT Report STF 40A90049. Trondheim, 1990.

**Larssen-90**

Christian Larssen: Multi-paradigm reasoning. ELAB-RUNIT Report 1990 (Forthcoming).

**Lenat-86**

Doug Lenat, M. Prakash, Mary Shepherd: CYC - using common sense knowledge to overcome brittleness and knowledge acquisition bottlenecks. *AI Magazine*, Winter 1986. pp. 65-85

**Minsky-75**

Marvin Minsky: A framework for representing knowledge. In P. Winston (ed.): *The psychology of computer vision*. McGraw-Hill, 1975. pp. 211-277.

**Murray-88**

Kenneth Murray, Bruce Porter: Developing a tool for knowledge integration; initial results. *Proceedings from the Third Knowledge Acquisition for Knowledge-Based Systems Workshop*, Banff, Canada, November 1988. 13 pgs.

**Nordbø-89**

Inge Nordbø, Mette Vestli, Ingeborg Sølvsberg: METAMETH; Methodology for knowledge engineering. In John Boose, Brian Gaines, Jean-Gabriel Ganascia (eds.): *EKAW-89; Third European Workshop on Knowledge-Based Systems*, Paris, July 1989.

**Porter-86**

Bruce Porter: PROTOS; an experiment in knowledge acquisition for heuristic classification tasks. *Proceedings of the First Intern. Meeting on Advances in Learning, Les Arcs*, France. July 1986. pp. 159-174. Also: UT, AI-Lab, Tech.report AI-TR-85-03. 1986.

**Quillian-68**

M. Ross Quillian: Semantic Memory. In Marvin Minsky (ed): *Semantic information processing*. MIT Press, Cambridge, 1968. pp. 216-260. Also in [Collins-85], pp. 79-101.

**Rissland-89**

Case-based reasoning, from DARPA Machine Learning Program Plan. By: Edwina Rissland, Janet Kolodner, David Waltz. In: *Proceedings from the Case-Based Reasoning Workshop*, Pensacola Beach, Florida, May-June 1989. Sponsored by DARPA. Morgan Kaufmann, 1989. pp 1-13.

**Schank-82**

Roger Schank: *Dynamic memory*. Cambridge University Press. 1982.

**Sølvsberg-88**

Ingeborg Sølvsberg, Inge Nordbø, Mette Vestli, Tore Amble, Geir Aakvik, Jorun Eggen, Agnar Aamodt: METAKREK - Methodology and tool-kit for knowledge acquisition. ELAB-RUNIT Report STF14 A88046. SINTEF, Trondheim. 1988.

**Thagard-88**

Paul Thagard: *Computational Philosophy of Science*. MIT Press/Bradford Books. 1988.

**Vestli-89**

Mette Vestli, Inge Romslo, Ingeborg Sølvsberg, Magnhild Prytz, Eli Kjølbi, Jorun Eggen, Harald Johnsen, Inge Nordbø, HJens Gleditch: METAKREK - A knowledge acquisition study. *Proceedings Expert systems and their applications, 9th international workshop*. Avignon, France, mai-juin 1989.