Chapter 7

Conclusions and future work

This thesis addressed the design and implementation of a representation language for developing knowledge systems that integrate problem solving and learning.

We have developed Noos, a reflective object-centered representation language for integrating inference and learning components in a uniform representation.

7.1 The Noos language and feature terms

Noos a representation language close to knowledge modeling frameworks, based on the task/method decomposition principle and the analysis of models required and models constructed by problem solving methods. This capability allows Noos to take advantage of the KA methodologies and libraries developed in KM frameworks.

The Noos modeling framework is based on four knowledge categories: domain knowledge, problem solving knowledge, episodic knowledge, and metalevel knowledge:

- Domain knowledge specifies a set of concepts, a set of relations among concepts, and problem data that are relevant for an application. Concepts and relations define the domain ontology of an application.

- Problem solving in Noos is considered as the construction of the *episodic model* of a problem. This model is obtained from transformations of problem data performed by problem solving knowledge. Episodic models built in solving problem tasks constitute the episodic knowledge of the system.

- Problem solving knowledge specifies a set of tasks and methods that construct a model of a problem (solve a problem). For a given subtask there may be multiple alternative methods that may be capable of solving that subtask in different situations. A method can be decomposed into subtasks that may be achieved by other methods.

- Metalevel knowledge specifies knowledge about domain knowledge, problem solving knowledge, and episodic knowledge. These models are formed by metalevel concepts, metalevel relations, metalevel tasks, and metalevel methods. Moreover, metalevel knowledge includes preferences to model decision making about sets of alternatives present in domain knowledge and problem solving knowledge.

Noos is an object-centered representation language based on *feature terms*. Feature terms are related to the research based on λN calculus [Dami, 1994] and *ψ-terms* [Aït-Kaci and Podelski, 1993] that propose formalisms to model relational and object-oriented programming constructs. Adapting the theoretical results of these formalisms to our purpose, feature terms provide a natural way to describe partial knowledge amenable to extension. Feature terms are interpreted as partial descriptions denoting sets of individuals in a given domain. This semantical interpretation of feature terms brings about an ordering relation among them. We call this ordering relation *subsumption*. The intuitive meaning of subsumption is that of informational ordering. We say that a feature term ψ subsumes another feature term ψ' when all information in ψ is also contained in ψ'—or in other words, ψ is more general than ψ'.

Noos incorporates a declarative mechanism for decision making about sets of alternatives called *preferences*. For instance, preferences are used as a declarative control mechanism for determining the order in which a metalevel task chooses a method for a task from a set of alternative methods.

Furthermore, preferences are also used in Noos as a symbolic representation of relevance in comparing a given current problem with problems previously solved by the system. Specifically, preferences are used in the retrieval and selection of precedent cases in case-based reasoning methods in applications such as CHROMA and *Saxex*.

Inference starts in Noos when the user poses a query to the system by means of a query expression. A query expression engages a task `F(D)` to be solved by its corresponding method. When solving a task where neither a path reference nor a method is defined, an impasse occurs and the control of the inference is passed to its corresponding metalevel task. Solving an impasse for a task `F(D)` involves three processes: (i) determining a set of methods $\{M_i\}_{F(D)}$ applicable to task `F(D)`, that can be partially ordered with preferences, (ii) selecting a method from $\{M_i\}_{F(D)}$, according to the preferences, and (iii) reflecting down the selected method to task `F(D)`.

Backtracking is engaged when a method fails in solving a task. In that case, another remaining non-failed method in $\{M_i\}_{F(D)}$ will be selected and reflected down. Moreover, since a method `M` can have subtasks, and each subtask may have several alternative methods to solve it, metalevel inference ensures that backtracking is engaged in `M`. Then, the possible combinations of methods for each subtask are tried, following the local preference orderings for each subtask, until a solution is found.

Methods in Noos can be view as functions with named parameters and backtracking. We formally described the global inference process in our system using Descriptive Dynamic Logic [Sierra et al., 1996], a propositional dynamic logic that provides a general framework for describing and comparing reflective knowledge systems.

7.2 Memory and learning

Our goal has been to provide a representation language for developing knowledge systems with learning capabilities. This goal required that machine learning techniques had to be modeled inside our language. Our proposal is that learning methods are methods (in the sense of knowledge modeling PSM) that can be analyzed also by means of a task/method decomposition and a set of models required models constructed by learning methods.

Both problem solving methods and learning methods perform inference (viewed in Noos as constructing episodic models). Learning methods differ from PSMs in that they use episodic models (i.e. past solved problems). Different kinds of learning methods use episodic models in different ways (see below the different approach of induction versus lazy learning approaches). These solved problems can be provided by a teacher or can be problems previously solved by the system itself. For learning methods to be able to reason from problems solved by the system, part of the behavior of the system has to be reified and stored in the system. In Noos, the episodic memory stores this representation of part of the behavior of the system. Moreover, we have incorporated a collection of reflective operations for accessing to and inspecting the episodic memory contents.

Episodic memory: Problems solved in Noos are automatically memorized (stored and indexed) in the episodic memory and are amenable to be accessed and reused in solving new problems. The problem solving behavior is represented in Noos in terms of tasks, methods, metalevels, and preferences. Episodic memory is organized in episodic models. An episodic model is the *explanation* of the inference process engaged by Noos in solving a specific problem task. An episodic model holds the set of knowledge pieces used for solving a specific problem task, how and where they were used, and the decisions taken for solving that problem task. Introspective capabilities form the basis that allows Noos programs to reason about the system behavior.

Integrated problem solving and learning: Our approach to integrate problem solving and learning is based on the following: whenever some knowledge is required by a problem solving method, and that knowledge is not directly available, there is an opportunity for learning. We call those opportunities *impasses* and the integration of learning is realized by learning methods that are capable of solving these impasses.

We have modeled the integration of different symbolic learning techniques as methods that can be decomposed in three main common subtasks: *Introspection*, *Construction*, and *Revision*. This common scheme allows us to model different ML methods and their integration into a general problem solving system by developing specific methods for the three main subtasks. Specifically:

- The introspection task is the process by which past experience (episodic memory of the system itself or provided by a teacher) is accessed, selected and retrieved for the purpose of solving new problems. In simple situations this task may merely select a subset of examples in memory. In complex situations the system may have to decide which (sub)parts of all the episodic memory qualify as "examples" (precedents), i.e. they are interesting to learn from.

- The construction task uses the relevant past experience (resulting from introspection) to generate some new model or body of knowledge.
  Eager learning methods construct a new model to be used for a specific problem solving method (that will be applied to future problems). Examples of eager learning methods are induction and analytical learning methods.
  Lazy learning methods follow a problem-centered approach—i.e. directly building the episodic model of a current problem from episodic model(s) of (some) retrieved precedent(s). Examples of lazy learning methods are CBR methods.

- The revision task decides whether and how the system's knowledge is modified by the newly constructed model. In simple situations the new model replaces the old model. In more complex situations, the task has to estimate whether the new model does improve the overall performance of the system.

Feature terms provide a structured representation of knowledge. A problem to be solved by the system is represented as a collection of concepts with many relationships among them. This structured representation of precedents offer the capability of treating subparts of them as full-fledged cases. That is to say, a new problem can be solved using subparts of multiple precedents retrieved from the episodic memory. This requires that new introspective mechanisms have to be provided. We have developed three introspective mechanisms to access the episodic memory: *access by path*, that provides an access to the episodic memory combining reflective operations and path references; *retrieval methods*, that provide a mechanism for content-based access to the episodic memory; and *perspectives*, a mechanism to describe declarative biases for case retrieval in the structured representation of cases.

Content-based retrieval: Since knowledge in Noos is represented in a structured way, retrieval methods have to deal with structured representations. Retrieval methods allow to retrieve previous relevant episodes from the episodic memory using relevance criteria. Relevance criteria are determined by specific domain knowledge about the importance of different features or by requirements of problem solving methods. Several domain-specific retrieval methods have been developed in Noos applications for solving the introspection task.

Perspectives: In complex tasks, the identification of the relevant aspects for retrieval in a given situation may involve the use of knowledge-intensive methods. This identification process requires dynamical decisions about the relevant aspects of a problem. Perspectives provide Noos with a mechanism for dynamically constructing retrieval patterns that specify the relevant aspects of a given problem. For instance, perspectives provide to GYMEL and *Saxex* applications a flexible way to dynamically construct partial descriptions for retrieval.

7.3 Methods and applications

Using the representation capabilities of Noos for modeling domain knowledge, problem solving knowledge, and learning, several PSMs and learning methods have been developed and integrated in several applications.

Based on the task/method decomposition principle, problem solving methods that use domain knowledge intensively can be designed and implemented. For instance, in the SHAM application several PSMs have been developed making an intensive use of background musical knowledge for harmonizing melodies. SHAM models musical knowledge by means of (1) PSMs that identify which notes are important in the melody, (2) PSMs that propose alternative sets of feasible chords according to these notes, and (3) PSMs that combine the alternatives constructing a complete chord sequence.

Furthermore, two complex musical theories for musical perception and musical understanding are modeled in *Saxex* and are then used for analyzing musical phrases and assessing their similitude with respect to other phrases in episodic memory.

Noos provides a collection of basic mechanisms allowing the development of different symbolic learning methods such as inductive learning, CBR, and analytical learning:

- *Inductive learning methods* in Noos are search methods (that follow certain biases) over the space of feature terms. Inductive learning methods are based on the feature term subsumption and antiunification operations of Noos. Subsumption provides a generalization relationship over feature terms. The antiunification of a set of feature terms builds a new feature term that is a greatest lower bound with respect to the subsumption ordering. Several strategies have been developed for constructing inductive learning methods that follow different searching biases.

- *Case-based reasoning methods* in Noos are problem solving methods with lazy learning capabilities that search for previously solved problems in the Noos episodic memory. CBR methods are based on the retrieval and subsumption operations of Noos.

Structured representations of cases offer the capability of treating subparts of cases as full-fledged cases. That is to say, a new problem can be solved using subparts of multiple cases retrieved from the episodic memory.
On the other hand, structured representations of cases increase the complexity of retrieval mechanisms. Noos provides elements—such as content-based retrieval and perspectives—for supporting the retrieval on these complex representations of cases.
Furthermore, derivational analogy is automatically supported by the Noos reinstantiation mechanism.

- *Analytical learning methods* in Noos are methods that given (1) a training example whose problem task has been solved by a problem solving method M and (2) an operationality criterion, they construct a new problem solving method M_{op} for solving that task and obeying the operationality criterion. Analytical learning methods are based on the Noos introspective capabilities for inspecting the methods used in subtasks of M for solving the training example.

Several persons at the IIIA have developed learning methods and integrated them to applications. For instance, several domain specific case-based reasoning methods have been developed in Noos and integrated to applications such as CHROMA, SPIN, GYMEL, and *Saxex*. Moreover, several inductive learning methods such as INDIE, DISC, and a lazy learning method (called LID) have been developed by Eva Armengol in Noos and also integrated to CHROMA and SPIN applications. Finally, an analytical method called PLEC has been also developed and presented in this thesis.

We also presented how different learning methods can be designed and integrated in a problem solving system. Specifically, the research work realized in [Armengol, 1997] provides examples of knowledge systems developed in Noos that integrate different learning methods such as case-based reasoning and inductive learning methods.

Noos has been implemented using Common Lisp and currently is running on several platforms—including a window-based graphical interface for the Macintosh version of Noos.

Finally, we want to remark that Noos has been used, and is also currently used, by several persons at the IIIA and by other people with collaboration with the IIIA to develop different applications that integrate several problem solving methods and several learning methods. Specifically, we have described five applications developed using Noos and a WWW interface to Noos applications supporting the same interaction capabilities provided in the Noos window-based graphical interface developed for MacOS computers:

- CHROMA, a system for recommending a plan for the purification of proteins from tissues and cultures using chromatographic techniques (developed by Eva Armengol);

- SPIN, a system for identifying specimens of *Geodiidae* marine sponge family (developed by Eva Armengol);

- SHAM, a system for assisting a non expert musician to harmonize melodies using background musical knowledge (developed by Martí Cabré);

- GYMEL, another system for assisting the harmonization of melodies that uses a case-based reasoning approach (developed by Jordi Sabater);

- *Saxex*, a case-based reasoning system for generating expressive performances of melodies based on examples of human performances that I have developed in the context of my M.Sc.Thesis in Computer Music (this application has been awarded with the "Swets & Zeitlinger Distinguished Paper Award" at the 1997 International Computer Music Conference); and

- NoosWeb, a WWW interface to Noos applications (developed by Francisco Martín).

7.4 Future work

From the work realized in this thesis several research lines appear to be sufficiently interesting to pursue. Some of them have already started with a preliminary results.

- A first research line is to extend the basic retrieval and subsumption mechanisms provided in Noos enriching the comparison on numbers and strings. The work of G. Kamp [Kamp, 1997] on the admissibility of concrete domains can be adapted to Noos for this purpose.

- Another extension of Noos is to provide an agent-based environment for the cooperation between different Noos applications and on an heterogeneous environment. We have already started the study of how different case-based reasoning agents can cooperate in solving problems. We are developing two modes of cooperation among CBR agents: *Distributed Case-based Reasoning* (DistCBR) and *Collective Case-based Reasoning* (ColCBR). Intuitively, in DistCBR cooperation mode an agent A_i *delegates its authority* to another *peer* agent A_j to solve a problem—for instance when A_i is unable to solve it adequately. In contrast, ColCBR cooperation mode *maintains the authority* of the originating agent: an agent A_i can transmit a mobile method to another agent A_j to be executed there. That is to say, A_i *uses the experience* accumulated by other peer agents while maintaining the control on *how* the problem is solved. These preliminary results has been presented on [Plaza et al., 1997] [Plaza et al., 1996a].

- A third research line is to explore and extend the Noos capabilities for learning from failure. The episodic model built in solving a problem task stores the collection of preference orders between alternative methods for solving a specific subtask and the method that has been effectively used in that subtask. Using this information we can infer the set of methods that has been failed in achieving the task. Nevertheless, Noos does not maintain the inference processes involved in the evaluation of these failed methods. This kind of knowledge could be useful for detecting situations where a PSM is not able to find a solution and then, to avoid failures in future problems.

- Another research line already started is focused on the study of the degradation of the system performance when the cost of searching for related knowledge outweighs the benefit of applying this knowledge. This problem is called the *utility problem* in [Tambe et al., 1990] and [Minton, 1990]. Different strategies have been proposed for solving this problem and could be useful also to Noos applications. Specifically, the research on deletion strategies in the context of case-based reasoning such as in [Smyth and Keane, 1995] is a promising direction to follow.

- Finally, a fifth research line already started is focused on developing new learning methods. Specifically, we are exploring inductive methods for acquiring methods from examples (as in inductive logic programming).