The AAAI Fall Symposium; Flexible Computation in Intelligent Systems: Results, Issues, and Opportunities. Nov. 9-11, 1996, Cambridge, MA

Supporting Flexibility. A Case-Based Reasoning Approach*

Igor Jurisica
Department of Computer Science
University of Toronto, Toronto, Ontario M5S 1A4
juris@cs.utoronto.ca

Abstract

This paper presents a case-based reasoning system $\mathcal{TA}$**3**. We address the flexibility of the case-based reasoning process, namely flexible retrieval of relevant experiences, by using a novel similarity assessment theory. To exemplify the advantages of such an approach, we have experimentally evaluated the system and compared its performance to the performance of non-flexible version of $\mathcal{TA}$**3** and to other machine learning algorithms on several domains.

Introduction

Flexible Computation

There are many situation when resources are scarce and when the system has to make decision as to how to proceed with further computation in order to make the right tradeoff between quality of the answer and resources needed. In several domains, such as medicine, robotics, financing, etc., if the answer to a query is not produced within a certain time limit it may become useless.

Such family of problems has been a motivation for designing anytime algorithms (Dean & Boddy 1988; Frisch & Haddawy 1994), algorithms based on decision-theoretic meta-reasoning under scarce and uncertain resources (Horvitz 1988; D'Ambrosio 1990) and algorithms that use time-dependent utility (Horvitz & Rutledge 1991; Haddawy & Hanks 1992; Wellman & Doyle 1992). Anytime algorithms allow for producing correct answer at any time, i.e., although the solution may improve by using more computational resources, intermediate results are usable as partial solutions. Horvitz's approach generates partial results characterized by multidimensional computational utility. Partial results represent transformations of the final result along certain dimension of the multidimensional utility. In addition, reasoner uses uncertainty about resources to weight its decisions. Utility approaches define an optimization function (utility measure) and optimization variables. Reasoner's options for next action are weighted and the action with the highest utility is pursued.

Case-Based Reasoning

Case-based reasoning systems rely on the concept that similar problems have similar solutions (Kolodner 1993). Thus, facing a new problem, a case-based system retrieves similar cases stored in a case base and adapts them to fit the problem at hand.

Informally, a case comprises an input (the problem), an output (the solution) and feedback (an evaluation of the solution). Case-based reasoning involves the process of: (1) Accepting a new problem description (a case without a solution and feedback) from a client (user or other part of the system). (2) Retrieving relevant cases from a case base (past problems with similar input). (3) Adapting retrieved cases to fit the problem at hand and producing the solution for the input problem. (4) Evaluating the solution (producing the feedback for the case).

Considering the case-based reasoning cycle, one can say that the more similar the cases are the less adaptation is necessary and consequently, the proposed solution may be more precise. Then, an important task is how to measure case closeness to guarantee retrieving only highly relevant cases, i.e., cases that are similar according to specified criteria and thus can be useful in solving the current problem. In many processes, it is better to retrieve fewer cases, or none, than to retrieve less useful cases that would result in a poor solution. In some situations, an accurate solution is required, with minimal concern on resources needed (Jurisica & Shapiro 1995). In other situations, time resources may be limited (e.g., during computing inverse kinematic task in robotics (Jurisica & Glasgow 1995)) and thus the system should allow for less accurate result but within time constraints.

This paper presents a prototype of a case-based reasoning system, called $\mathcal{TA}$**3**, that allows for flexible retrieval of cases. Flexibility is needed to support the

*This research was supported by the Information Technology Research Centre of Ontario. Author is indebted to J. Glasgow, J. Mylopoulos for helpful comments and suggestions.

following aspect of $\mathcal{TA}$**3**:

- Variable accuracy of the final solution as a function of resources used.
- Changeable retrieval objective: recall-oriented retrieval (all relevant cases should be retrieved, even if some irrelevant are included) and precision-oriented retrieval (only relevant cases must be retrieved).
- Imprecise query specification to allow for retrieving cases even if the query is not specified completely and retrieving similar cases in addition to exact matches.
- Automatic query modifications (restriction and relaxation) as well as query-by-example and query-by-reformulation.

These features are needed for supporting a variety of different domains and tasks since the task requirements may change over time and individual users may have different preferences. In addition, the retrieval must be efficient, i.e., the system must scale up as more cases are acquired.

The $\mathcal{TA}$3 System

$\mathcal{TA}$**3** is a case-based reasoning system which uses context-based relevance measure during similarity-based retrieval (Jurisica 1994). In $\mathcal{TA}$**3**, it is assumed that cases are represented as structured objects with relations (Mylopoulos *et al.* 1990). Objects or relations are sets of attributes and their values:

$$\mathcal{C} = \{< a_0 : V_0 >, < a_1 : V_1 >, \cdots, < a_n : V_n >\} = $$
$$= \{< a_i : V_i >\}_0^n = \{A_i\}_0^n,$$

where $A_i =< a_i : V_i >$ is an attribute/value pair, a_i is an attribute and V_i is a value (possibly a complex value). Two case types are distinguished:

- a source case $\mathcal{C}_{source}$ is a case in a case base, i.e., a case with a solution and
- an input case $\mathcal{C}_{input}$ (also referred to as a problem or a target) is a case that describes a given problem either in full or only partially, i.e., only subset of all attributes may be filled with values.

A case base, $\Delta = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_k\}$, is a finite set o cases – the search space of source cases.

Using the information about the usefulness of individual attributes and its properties attributes are grouped into one or more Telos-style categories (Mylopoulos *et al.* 1990). Each case can have one or more categories and the category membership is based on the usefulness of the attribute and its properties. This grouping allows for ascribing different constraints to different groups of attributes and the process of retrieving relevant cases can be described as a constraint satisfaction process (Thagard *et al.* 1990; Gaasterland 1993; Kokeny 1995). Deploying `Telos` (Mylopoulos *et al.* 1990) representation language allows for treating objects (cases) and their attributes uniformly and thus diminishing problems discussed earlier (Dierbach & Chester 1991). In addition, it allows for lowering the impact of irrelevant attributes on system performance (Aha 1992; Ortega 1995). Categories allow for improved system performance (Jurisica & Glasgow 1995; Jurisica & Shapiro 1995; Jurisica & Glasgow 1996a). Knowledge discovery techniques are used to organize case attributes into categories (Jurisica 1996; Jurisica *et al.* 1996).

The main goal is to retrieve only useful cases, i.e., cases highly relevant to the user's request. All cases that are relevant to a given request can be treated as equivalent. The retrieval component of the system is based on a nearest-neighbor matching: using a case base, the system locates cases similar to the problem description. This is, however, different from retrieval in traditional information retrieval systems. In such systems it is assumed that the user knows exactly what (s)he is searching for. In our system, it is assumed that the problem is only a partially developed description and that the task of the system is to find not only exact matches, but similar cases as well. $\mathcal{TA}$**3** uses a novel theory of relevance measure, where case relevance in a particular situation is determined by using context and similarity (Jurisica 1994; Jurisica & Glasgow 1996a). By changing the context, one can change the relevance measure flexibly.

Relevance of cases to a given request can be measured using various approaches. However, there is little commonality between definitions of relevance used in different systems (Greiner 1994), except when relevance is defined as usefulness. Since the usefulness of individual attributes varies, we define context as a set of attributes, relevant for a given retrieval. Thus, the context can be seen as a view or an interpretation of a case, where only subset of all attributes are considered relevant. Context can be formed by using only certain attributes of a target case and the similarity is determined as a closeness of values for given attributes (note that closeness depends on attribute value distribution as well as on the task being solved). Formally, a context, Ω, is defined as a finite set of attributes with associated constraints on the attribute values:

$$\Omega = \{< a_0 : CV_0 >, ..., < a_k : CV_k >\} =$$
$$= \{< a_i : CV_i >\}_0^k,$$

where a_i is an attribute name and constraint CV_i specifies the set of "allowable" values for attribute a_i. Context specifies important attributes and how "close" attribute value must be.

Context allows for specifying a variety of situations, ranging from completely ignoring the value of an attribute, through considering only certain ("allowed") values for an attribute, to considering a certain set, consisting of all values except some of "prohibited" values. This flexibility in context specification can be summarized as follows:

- ignoring an attribute during similarity assessment;

- using an attribute during similarity assessment (this is equivalent to specifying a null value in context, i.e., there is no constraint on the attribute value; thus, every value would match);
- using an attribute during similarity assessment with specific constraints on its values:
  - vaguely defining attribute's value (e.g., specifying domain, interval);
  - listing prohibited/allowed values;

If the interpretation of the target case (the problem) is similar to a source case (case in the case base), we say that they match (in other words, the case satisfies constraints defined in the context) and thus are relevant in a given context. Formally, a case $\mathcal{C}$ **satisfies** a context Ω, denoted $sat(\mathcal{C}, \Omega)$, if and only if for all pairs $< a_i : CV_i > \in \Omega$, there exists a pair $< a_i : V_i > \in \mathcal{C}$ such that V_i is in CV_i:

$$sat(\mathcal{C}, \Omega) \ \mathit{iff} \ \forall (< a_i : CV_i > \in \Omega \ \rightarrow$$
$$\exists (< a_i : V_i > \in \mathcal{C} \ \wedge \ V_i \in CV_i)).$$

We say that the context Ω is **satisfiable** if and only if there exist at least one case in a case base that satisfies it.

In addition, the theory allows for imposing constraints on attribute values, on category cardinality and on the schema:

1. **Value constraints** specify constraints on attribute values. *Instance* specifies that the attribute value in the source case must match the attribute value in the interpretation of the target case. *Range* requires that the attribute value in the source case falls within a specified range defined in the context. For a *domain* value constraint, the attribute value in the source case must be an instance of the attribute value in the context.
2. **Cardinality constraints** specify the number of attributes required to match for a particular category, similarly as m_of_n matching in (Ortega 1995). *Exact* specifies that all attributes in the category of a source case and the interpretation of the target case must match. *Some(m)* determines that not all attributes in a category are required to match, but at least m_of_n attributes must match the context. It should be noted that $Exact = Some(n)$.
3. **Schema constraints** are used for schema integration (Kashyap & Sheth 1993) and analogical inference (Vosniadou & Ortony 1989). Two matching strategies are supported – AttributeMode and MapMode. The first requires that both attribute name and attribute value match; the latter allows for matching if only the value matches.

By selecting only certain attributes for matching and imposing constraints on attribute values, context allows for controlling what can and what cannot be considered as a partial match: All cases that satisfy constraints specified in the context (i.e., for all attributes specified in the context, there is a match in values between target and a source case) are considered similar and are relevant with respect to a given context. This allows for a more controlled retrieval process, as well as for easy context modifications.

This notion of context subsumes traditional nearest-neighbor or template matching in CBR systems (Kolodner 1993), where the partial description of a case is submitted as the query and the system locates all cases similar to a given description. Our approach allows for initiating the query by using query-by-example; however, using context allows for subsequent controlled modifications. Besides specifying a context, a user can state criteria which define the way the context is handled and what criteria should be used for matching during context relaxation and restriction. Thus, context permits easy implementation of retrieval by reformulation (Devanbu *et al.* 1991) and allows for iterative browsing through contextually similar cases. When the problem of specifying query gets hard, retrieval by reformulation is the best approach, which is the motivation for supporting efficient iterative browsing.

The obvious advantage of an explicit context definition is not practical, if there is no easy way for specifying the context. However, there are four main scenarios available for defining a context:

1. The user has enough domain knowledge to specify the context (query-by-example can be used as a starting point in context specification).
2. The user specifies the task (s)he wants to solve and the system selects appropriate context (task-based retrieval).
3. First, a knowledge discovery algorithm is used to locate relevant attributes and attribute values. Then, a user forms a context for retrieval using this information. This approach is suitable for novice users.
4. The user submits an ad hoc query, reviews the resulting solution, and then iteratively modifies and resubmits the query (retrieval by reformulation). This approach is best suited for case base browsing.

Context can be modified by relaxing or restricting constraints. Relaxation can be performed either by search space expansion or by attribute constraints reduction. Expanding the search space can be implemented by enlarging the set of cases to be considered. Attribute constraints reduction is achieved either by relaxing the restriction on the number of relevant attributes of the source cases that must match with the target (cardinality relaxation) or by expanding the value set for a given attribute (value relaxation). Category priority (i.e., its weight) is used to decide which attributes to relax, giving the preference for categories with lower priority. If no priority is explicitly assigned then the last category is assumed to be the least important. Category priorities are defined using either the knowledge discovery algorithm or are specified by the user.

As pointed out in (Gaasterland 1993), the relaxation technique can advantageously be used for returning answer to a specific query as well as returning related answers. Without such automatic relaxation, users would need to submit alternative queries themselves. The restriction technique works analogously, but is used mainly for controlling the amount of returned information, keeping it in manageable levels for the user. Since the search for relaxed or restricted contexts can be infinite, there must be a mechanism for contorlling it, either by user intervention via user preferences or by other means.

Cardinality is relaxed from $Some(m)$ to $Some(m-1)$, for $m \geq 1$. If the value constraint is specified by enumeration of values then the expanded set of values relaxes the value constraint. On numeric attributes, this can be done easily. Domain knowledge in the form of concept hierarchies can be utilized during relaxation of numeric as well as symbolic attributes. Using such generalization hierarchies value constraint can be relaxed from an *instance* to a *domain*. When automatic expansion of the search space is selected, the algorithm finds super-classes for each class of the search space, selectively merges them, leaving out everything that was already checked, and then starts the retrieval again. Thus, relevancy-based retrieval can change the metrics on which relevance assessment is based. Context relaxation could be done either by the user or by the system.

Similar to relaxation – contexts can be iteratively restricted to retrieve successively fewer cases. Restricting a context means making it more specific. A restriction function takes context, restricts it according to criteria and returns a modified context. Either cardinality or value restriction can be applied, depending on criteria. During the restriction, the category with the highest priority is restricted first. If the restriction does not yield the expected result (i.e., if the quality of returned cases is not improved or quantity reduced), the cardinality with the second highest priority is used, etc. Only one category is restricted at a time.

Formally, a context Ω_1 is a **restriction** of a context Ω_2, denoted $\Omega_1 \preceq \Omega_2$, if and only if: for all (n) pairs $< a_i : CV_i > \in \Omega_1$, if there exists a pair $< a_j : CV_j > \in \Omega_2$ such that $a_i = a_j$ then $CV_i \subseteq CV_j$:

$$\Omega_1 \preceq \Omega_2 \; iff \; \forall < a_i : CV_i > \in \Omega_1,$$

$$\exists < a_j : CV_j > \in \Omega_2 \; : \; a_i = a_j \wedge CV_i \subseteq CV_j.$$

Conversely, if Ω_1 is a restriction of Ω_2 then we say that Ω_2 is a **relaxation** of Ω_1, denoted $\Omega_2 \succeq \Omega_1$

$$\Omega_2 \succeq \Omega_1 \; iff \; \forall < a_i : CV_i > \in \Omega_2,$$

$$\exists < a_j : CV_j > \in \Omega_1 \; : \; a_i = a_j \wedge CV_i \supseteq CV_j$$

Performance Evaluation

In (Jurisica 1995b) we have discussed methods that can be used for evaluating retrieval systems intended for software repositories. The main objective of our evaluation was to test the task performance, i.e., the ability of $\mathcal{TA}$**3** to solve a given task: in a medical domain – suggesting the hormonal therapy (Jurisica & Shapiro 1995; Jurisica *et al.* 1996), in a robotic domain – predicting the rise time of a servomechanism and solving inverse kinematic task (Jurisica & Glasgow 1995; 1996b), in a character recognition domain – recognizing scanned capital letters (Jurisica & Glasgow 1996a) and in a software engineering domain – supporting similarity-based retrieval of software artifacts (Jurisica 1995b). Afterwards, the evaluation concentrated on the scalability issues, namely on evaluating the incremental context modification algorithm used during case base browsing (Jurisica & Glasgow 1996b; Jurisica, Mylopoulos, & Glasgow 1996). Advantages of the knowledge discovery component were explained and evaluated in (Jurisica *et al.* 1996; Jurisica 1996).

Using different contexts and criteria affects the time spent on searching for a solution as well as its quality. $\mathcal{TA}$**3** allows for tuning up the reasoning process to meet different requirements. On the one hand, in the medical domain (Jurisica & Shapiro 1995; Jurisica *et al.* 1996), more accurate suggestion of the hormonal therapy has positive impact on pregnancy and it is also cost effective, since it minimizes the amount of hormones given to the patient. Thus, even though the treatment should be suggested reasonably fast, clearly the accuracy is more important. This can also be seen as an optimization problem: the optimization function is minimizing the cost of treatment while preserving high utility, i.e., pregnancy. The optimization variables include number of days hormones are administered and the used dosage. Depending on the patient's characteristics (age, diagnosis of infertility, previous treatment results, etc.) and the patient's response to the current treatment plan, it is possible to decide whether to continue with an expensive treatment or not, based on the probability of success. If the belief in success is smaller than a threshold value t then the patient should be warned of lower than usual probability of successful treatment.

On the other hand, in the robotic domain (Jurisica & Glasgow 1995; 1996b), time resources are limited. Thus, even approximate solution provided in time has greater utility than an accurate solution delivered late. There are fast techniques available to produce precise solutions from an approximate one for an inverse kinematic task. However, for some robot architectures, there might not be a computational solution to the problem. Thus, the main objective is to have any solution available within time constraints.

Retrieval from information repositories works similarly (Jurisica 1995b; 1995a). One may want to pursue recall-oriented retrieval and thus retrieving all relevant cases, even at the cost that some irrelevant cases are included. In other situation, precision-oriented retrieval may be desired and thus no irrelevant cases are to be retrieved, even at the cost of missing some relevant cases. Many information retrieval systems have the retrieval strategy predefined. $\mathcal{TA}$**3** not only allows for changing the strategy for different users or in different situations, it allows to change the strategy flexibly on-line. In other words, if the user is not an expert in the domain, then recall-oriented retrieval is the most useful strategy to start with, followed by incremental precision-oriented retrieval. The first step is used to retrieve all relevant cases, while the second step reduces the number of irrelevant cases. This approach has a positive impact on performance: the recall-oriented retrieval uses fewer constraints and thus is computationally simple. The problem of large number of retrieved cases is solved by the second step of precision-oriented retrieval. The second step introduces new constraints and thus is more complex, but is performed on a substantially smaller number of cases.

Evaluations show that the utility of $\mathcal{TA}$**3** can be custom-tailored for specific applications. Within a particular application model, another level of flexibility if supported via query manipulations and transformations.

Conclusions

Context-based similarity supports flexibility by allowing for different relevancy measures, different retrieval strategies, automatic context relaxation/restriction and by supporting similarity-based queries, query-by-example and query-by-reformulation. Flexible context modifications allow for controlling precision of the answer (Jurisica & Glasgow 1995; Jurisica & Shapiro 1995; Jurisica & Glasgow 1996a) and for controlling the quantity and quality of retrieved cases (Jurisica 1995b; Jurisica & Shapiro 1995). Thus, $\mathcal{TA}$**3** can be tuned for a particular application or a specific user.

Using the terminology of normative rationality introduced in (Good 1952), medical application of the $\mathcal{TA}$**3** system follows type I rationality, i.e., inference that is consistent with the axioms of decision theory regardless of the cost of inference. Robotic application of $\mathcal{TA}$**3** implements type II rationality, where the cost of reasoning is considered.

Flexible systems require strategies that continuously refine the value of partial results with time. $\mathcal{TA}$**3** supports this by context relaxation/restriction. Namely, based on obtained results (even partial) the system changes m in m_of_n matching to increase/reduce the number of retrieved relevant cases. By controlling this process, $\mathcal{TA}$**3** allows for improving retrieval precision while preserving perfect recall.

Flexibility of the $\mathcal{TA}$**3** system is further enhanced by incorporation of the knowledge discovery algorithm that allows for finding optimal knowledge representation from two points of view: precision of results and efficiency of computation. Thus, importance weight of attributes may dynamically change, which in turn changes retrieval strategies.

$\mathcal{TA}$**3** is different from other case-based approaches since instead of using predefined retrieval strategies, case retrieval is custom-tailored and dynamically changed for a particular domain and specific application. $\mathcal{TA}$**3** can be configured for individual tasks easily and allows for flexibility in certain aspects of computation, such as trading off the accuracy or precision of the computation process for time resources.

Although presented similarity-based retrieval is part of the case-based reasoning system $\mathcal{TA}$**3**, the same principle is applicable for any kind of retrieval, where repository items can be represented as structured objects with relations. Thus, the system can be used as a building block of flexible retrieval systems.

References

Aha, D. 1992. Tolerating noisy, irrelevant and novel attributes in instance-based lear ning algorithms. *International Journal of Man - Machine Studies* 36(2):267–287.

D'Ambrosio, B. 1990. Process, structure, and modularity in reasoning with uncertainty. In Shachter, R. D.; Levitt, T. S.; Kanal, L. N.; and Lemmer, J. F., eds., *Uncertainty in Artificial Intelligence 4*, 15–25. Amsterdam: North-Holland.

Dean, T., and Boddy, M. 1988. An analysis of time-dependent planning. In *Proc. of AAAI-88*, 49–54.

Devanbu, P.; Ballard, B.; Brachman, R.; and Selfridge, P. 1991. LaSSIE: A knowledge-based software information system. In Lowry, M., and McCartney, R., eds., *Automatic software design.* Cambridge: AAAI Press/MIT Press.

Dierbach, C., and Chester, D. L. 1991. A formal basis for analogical reasoning. In *Proc. of 2^{nd} International Conference on Principles of Knowledge Representation and Reasoning.*

Frisch, A. M., and Haddawy, P. 1994. Anytime deduction for probabilistic logic. *Artificial Intelligence* 69:93–122.

Gaasterland, T. 1993. Restricting query relaxation through user constraints. In *Proc. on International Conference on Intelligent and Cooperative Information Systems*, 359–366.

Good, I. 1952. Relational decisions. *J. R. Stat. Soc. B* 14:107–114.

Greiner, R. 1994. *AAAI Fall Symposium Series on Relevance.* Menlo Park, CA: AAAI Press.

Haddawy, P., and Hanks, S. 1992. Representations for decision-theoretic planning: Utility functions for deadline goals. In Nebel, B.; Rich, C.; and Swartout, W., eds., *Principles of Knowledge Representation and Reasoning: Proc. of the Third International Conference*, 71–82.

Horvitz, E., and Rutledge, G. 1991. Time-dependent utility and action under uncertainty. In D'Ambrosio, B. D.; Smets, P.; and Bonissone, P. P., eds., *Uncertainty in Artificial Intelligence: Proc. of the Seventh Conference*, 151–158. San Mateo, CA: Kaufmann.

Horvitz, E. J. 1988. Reasoning under varying and uncertain resource constraints. In *Proc. of AAAI-88*, 111–116.

Jurisica, I., and Glasgow, J. 1995. Applying case-based reasoning to control in robotics. In 3^{rd} *Robotics and Knowledge-Based Systems Workshop.*

Jurisica, I., and Glasgow, J. 1996a. Case-based classification using similarity-based retrieval. In 8^{th} *IEEE International Conference on Tools with Artificial Intelligence.*

Jurisica, I., and Glasgow, J. 1996b. A case-based reasoning approach to learning control. In 5^{th} *International Conference on Data and Knowledge Systems for Manufacturing and Engineering.*

Jurisica, I., and Shapiro, H. 1995. A computer model for case-based reasoning in IVF. In *The* 51^{st} *Conference of the American Society for Reproductive Medicine.*

Jurisica, I.; Mylopoulos, J.; Glasgow, J.; Shapiro, H.; and Casper, R. F. 1996. Case-based reasoning in IVF: Prediction and knowledge mining. *AI in Medicine. Special issue on CBR in Medicine.* Under revisions.

Jurisica, I.; Mylopoulos, J.; and Glasgow, J. 1996. Performance analysis of an incremental algorithm for the $\mathcal{TA}$**3** CBR system. Submitted.

Jurisica, I. 1994. How to retrieve relevant information? In *(Greiner 1994)*, 101–104.

Jurisica, I. 1995a. $\mathcal{TA}$**3** : Case-based intelligent retrieval and advisory tool. In *ACM Conference on Society and the Future of Computing.*

Jurisica, I. 1995b. A similarity-based retrieval tool for software repositories. In *The Third Workshop on AI and Software Engineering: Breaking the Mold. IJCAI-95.*

Jurisica, I. 1996. Inductive learning and case-based reasoning. In *Canadian AI Conference, Workshop on What is inductive learning?*

Kashyap, V., and Sheth, A. 1993. Schema correspondences between objects with semantic proximity. Technical Report DCS-TR-301, Department of Computer Science, Rutgers University.

Kokeny, T. 1995. Constraint satisfaction problems with order-sorted domains. *International Journal on Artificial Intelligence Tools* 4(1 & 2):55–72.

Kolodner, J. L. 1993. *Case-Based Reasoning.* San Mateo, CA: Morgan Kaufmann.

Mylopoulos, J.; Borgida, A.; Jarke, M.; and Koubarakis, M. 1990. Telos: Representing knowledge about information systems. *ACM Transactions on Information Systems* 8(4):325–362.

Ortega, J. 1995. On the informativeness of the DNA promoter sequences domain theory. *Journal of Artificial Intelligence Research* 2:361–367. Research Note.

Thagard, P. R.; Holyoak, K. J.; Nelson, G.; and Gotchfeld, D. 1990. Analog retrieval by constraint satisfaction. *Artificial Intelligence* 46:259–310.

Vosniadou, S., and Ortony, A. 1989. *Similarity and Analogical Reasoning.* New York, NY: Cambridge University Press.

Wellman, M. P., and Doyle, J. 1992. Modular utility representation for decision-theoretic planning. In Hendler, J., ed., *AIPS-92: Proc. of the First International Conference on Artificial Intelligence Planning Systems*, 236–242.