Specifying and Learning Inductive Learning Systems Using Ontologies

Akihiro SUYAMA and **Takahira YAMAGUCHI**
School of Information, Shizuoka University
3-5-1 Johoku Hamamatsu Shizuoka, 432-8011 JAPAN
{suyama, yamaguti}@cs.inf.shizuoka.ac.jp

Abstract

Here is presented a platform for automatic composition of inductive learning systems using ontologies called CAMLET, based on knowledge modeling and ontologies engineering technique. CAMLET constructs an inductive learning system with better competence to a given data set, using process and object ontologies. Afterwards, CAMLET instantiates and refines a constructed system based on the following refinement strategies: greedy alteration, random generation and heuristic alteration. Using the UCI repository of ML databases and domain theories, experimental results have shown us that CAMLET supports a user in constructing a inductive learning system with best competence.

Introduction

During the last ten years, knowledge-based systems (KBSs) have been developed using knowledge modeling techniques. In particular, in order to exploit reusable knowledge components, extensive research effort has been placed on exploiting problem solving methods (PSMs) at high levels of abstraction, such as Generic Tasks (T.Bylander et al., 1987), PROTEGE-II (Musen et al., 1992) and Common-KADS (J.Breuker et al., 1994). PSMs are high-level languages specify problem solving processes independent of implementation details. Now the research effort moves into ontologies engineering, together with PSMs. An ontology is an explicit specification of a conceptualization (Gruber, 1992). According to (Gertjan van Heijst, 1995), there are several distinguished ontologies, such as generic ontologies for conceptualizations across many domains, domain ontologies to put constraints on the structure and contents of domain knowledge in a particular-field, and PSMs (some researchers call them task ontologies recently).

On the other hand, during the last twenty years, many inductive learning systems, such as ID3 (J.R.,Quinlan, 1986), GA based classifier systems (L.B.Booker et al., 1989) and data mining systems, have been developed, exploiting many inductive learning algorithms. However, the competence with inductive learning systems changes, depending on the characteristics of given data sets. So far we have no powerful inductive learning systems that always work well to any data set.

From the above background, it is time to decompose inductive learning algorithms and organize inductive learning methods (ILMs) for reconstructing inductive learning systems. The competence of ILMs changes depending on such properties of given data set as liner discrminability or non-liner dicrminability and much or less noise. Given such ILMs, we may construct a new inductive learning system that works well to a given data set by reinterconnecting ILMs. The issue is to learn (or search) a inductive learning system good for a given data set. Thus this paper focuses on specifying ILMs into an ontology for objects manipulated by learning processes (called a process ontology here) and also an object ontology for objects manipulated by learning processes. After constructing two ontologies, we design a computer aided machine (inductive) learning environment called CAMLET and evaluates the competence of CAMLET using several case studies from UCI Machine Learning Repository.

Ontologies for Inductive Learning

Before specifying ontologies for inductive learning processes, we have analyzed popular inductive learning systems, such as ID3 (J.R.,Quinlan, 1986), GA based classifier systems (L.B.Booker et al., 1989) and data mining systems. A process ontology is for ILMs that compose inductive learning systems. An object ontology is for objects manipulated by ILMs from the process ontology. In order to specify process and object ontologies, we need to specify conceptual hierarchies and conceptual schemes (definitions) on two ontologies.

Process Ontology

In order to specify the conceptual hierarchy of a process ontology, it is important to identify how to branch down processes. Because the upper part is related with general processes and the lower part with specific processes, it is necessary to set up different ways to branch the hierarchy down, depending on the levels of hierarchy.

In specifying the upper part of the hierarchy, we have analyzed popular inductive learning systems and then identified the following five popular and abstract components : "generating training and test data sets", "generating a classifier set", "evaluating data and classifier sets", "modifying a training data set" and "modifying a classifier set", with the top-level control structure as shown in Figure 1. Although we can place finer components on the upper part, they seem to make up many redundant composition of inductive learning systems. Thus these five processes have been placed on upper part in the conceptual hierarchy of the process ontology, as shown in Figure 2.

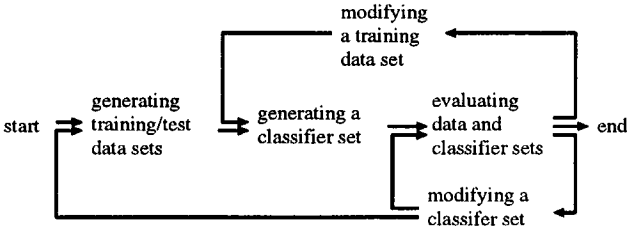

Figure 1: Top-level Control Structure

In specifying the lower part of the hierarchy, the above abstract component has been divided down using characteristics specific to each. For example "generating a classifier set" has been divided into "(generating a classifier set) dependent on training sets" and "(generating a classifier set) independent of training sets" from the point of the dependency on training sets. Thus we have constructed the conceptual hierarchy of the process ontology, as shown in Figure 2. Furthermore, the division puts restrictions on control structure. For example, when "generating a classifier set" comes up after "modifying a training data set", "generating a classifier set depend on a training set" is valid, but "generating a classifier set independence from a training set" is invalid. In Figure 2, leaf nodes correspond to the library of executable program codes that have been manually developed by C language.

On the other hand, in order to specify the conceptual scheme of the process ontology, we have identified the learning process scheme including the following roles: "input", "output" and "reference" from the point of objects manipulated by the process, and then "pre-process" just before the defined process and "post-process" just after the defined process from the point of processes relevant to the defined process. In order to keep valid inductive learning systems constructed by CAMLET, CAMLET needs much more relationships among process ontology components but it has not yet been done.

Object Ontology

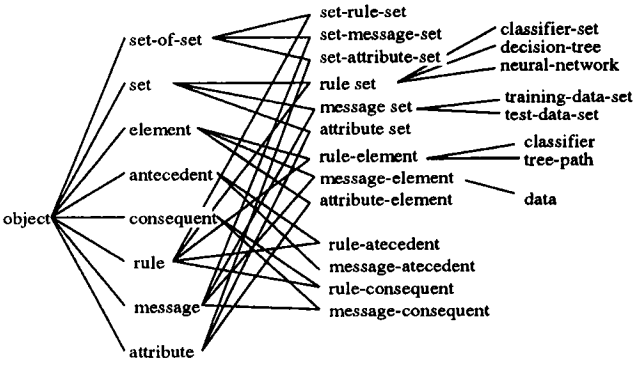

Figure 3: Hierarchy of Object Ontology

In order to specify the conceptual hierarchy of the object ontology, we use the way to branch down

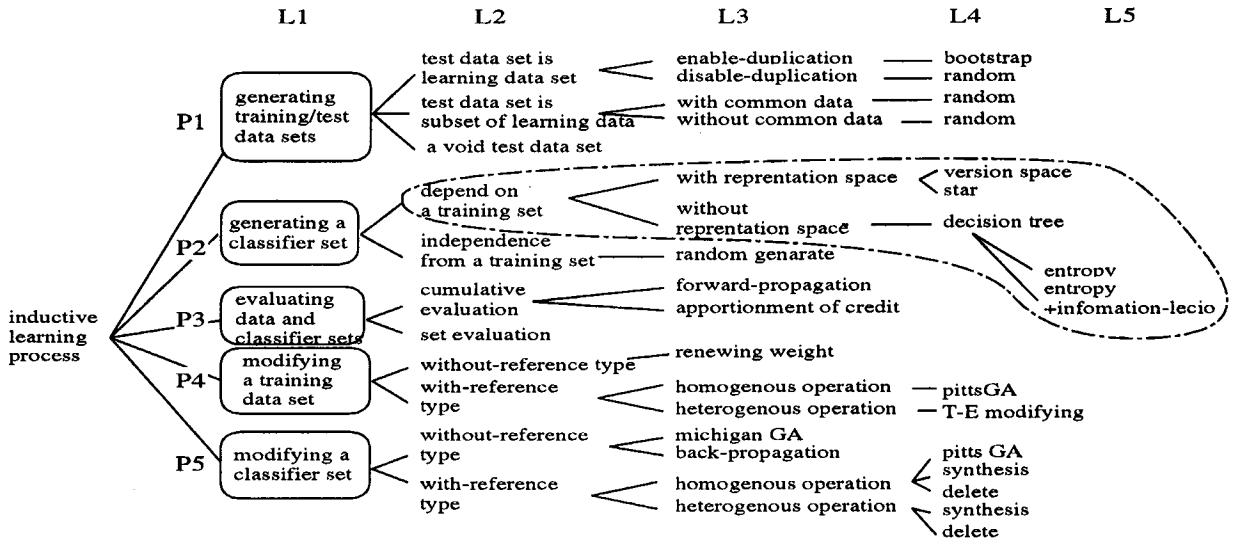

Figure 2: Hierarchy of Process Ontology

the data structures manipulated by learning processes, such as sets and strings, as shown in Figure 3. Because objects contribute less to construct inductive learning systems than processes, object scheme has less information than process. So it has just one role "process-list" that is a list of processes manipulating the object.

Basic Design of CAMLET

Figure 4 shows the basic activities for knowledge systems construction using PSMs (Gertjan van Heijst, 1995). In this section, we apply the basic activities to constructing inductive learning systems using process and object ontologies.

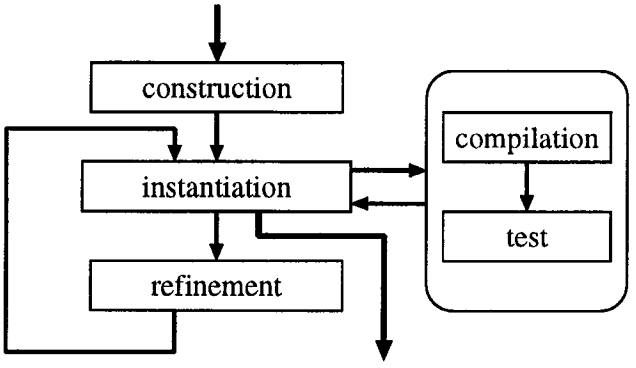

Figure 4: Basic Activities

The construction activity constructs an initial specification for an inductive learning system. CAMLET selects a top-level control structure for an inductive learning system by selecting any path from "start" to "end" in Figure 1. Afterwards CAMLET retrieves leaf-level processes subsumed in the selected top-level processes, checking the interconnection from the roles of pre-process and post-process from the selected leaf-level processes. Thus CAMLET constructs an initial specification for an inductive learning system, described by leaf-level processes in process ontology. In order to reconstruct the specification later, the selected leaf-level processes have been pushed down into a process stack.

The instantiation activity fills in input and output roles of leaf-level processes from the initial specification, using data types from a given data set. The values of other roles, such as reference, pre-process and post-process, have not been instantiated but come directly from process schemes. Thus an instantiated specification comes up. Additionally, the leaf-level processes have been filled in the process-list roles of the objects identified by the data types.

The compilation activity transforms the instantiated specification into executable codes using library for ILMs. When the process is connected to another process at implementation details, the specification for I/O data types must be unified.

To do so, this activity has such a data conversion facility that converts a decision tree into classifier.

The test activity tests if the executable codes for the instantiated specification goes well or not, checking the requirement (accuracy) from the user. The evaluation will come up to do a refinement activity efficiently, which is explained later. This activity evaluates how are good a top-level control structure in Figure 1 and four sub-control structures in Figure 5.

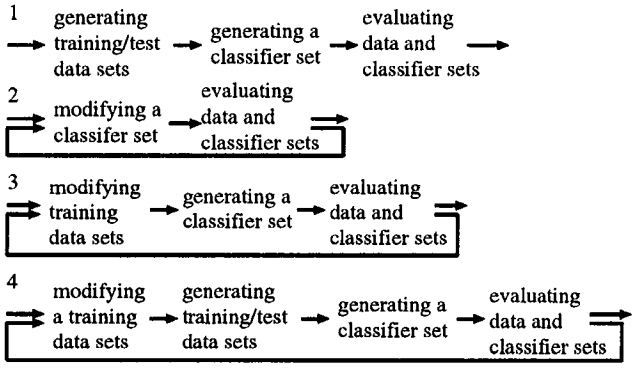

Figure 5: Sub-Control Structures

When the executable codes do not go well, the refinement activity comes up in order to refine or reconstruct the initial specification and get a refined specification back to the instantiation activity. The refinement activity is a kind of search task for finding out the system (or control structure) satisfied with a goal of accuracy. Although several search algorithms have been proposed, genetic programming (GP) is popular for composing programs automatically. GP goes well for global search but no so well for local search. So, in order to solve this problem, here is presented the hybrid search that combines GP with a local search with several heuristics based on empirical analysis. This activity has been done with the following three strategies: greedy alteration, random generation and heuristic alteration.

Greedy alteration makes a new system from two parent systems. This operation works like G-crossover in GP. Because evaluation values are added to sub-control structures at test activity, CAMLET can identify just sub-control structures with better evaluation values from parent systems and then put them into one new child system, as

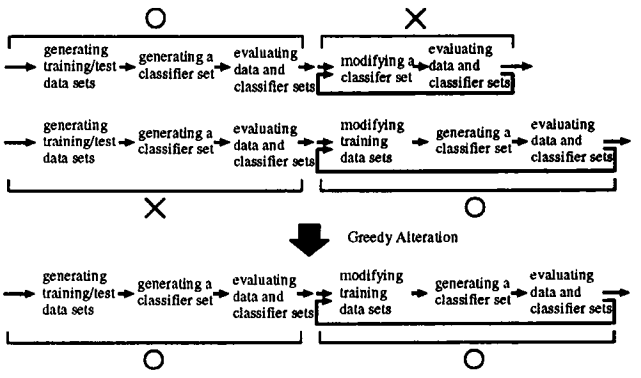

Figure 6: Greedy Alteration

shown in Figure 6.

Random generation makes a system with a new top-level control structure in the same way as construction activity. This activity is done, keeping various control structures in population.

Heuristic alteration change sub-control structures using several heuristics, which replace one process in a sub-control structure with another process from process ontology based on evaluation results. Heuristic alteration is a kind of local search.

Figure 7 summarizes the above-mentioned activities. A user gives a data set and a goal of accuracy to CAMLET. CAMLET constructs the specification for an inductive learning system, using process and object ontologies. When the specification does not go well, it is refined into another one with better performance by greedy alteration, random generation and heuristic alteration. To be more operational, in the case of a system's performance being higher than $\delta(= 0.7*$ goal accuracy), the heuristic alteration comes up. If not so, in the case of that system population size is equal or larger than some threshold ($N \geq \tau = 4$), CAMLET executes greedy alteration, otherwise, executes random generation. All the system refined by three strategies get into a system population. As a result, CAMLET may (or may not) generate an inductive learning system satisfied with the accuracy from the user. When it goes well, the inductive learning system can learn a set of rules that work well to the given data set.

Case Studies and Discussions

Based on the basic design, we have implemented

Table 1: Comparison of CAMLET and Popular Inductive Learning Systems

Data	C4.5	ID3	CS	B_C4.5	CAMLET	
	err (%)	err (%)	err (%)	err (%)	err (%)	goal (%)
annealing	9.00	18.00	21.00	11.00	9.00	10.00
audiology	26.92	15.38	84.62	11.54	11.54	12.00
breast-w	7.18	7.46	20.44	7.46	5.36	8.00
credit-s	18.48	20.38	24.18	16.85	13.53	15.00
glass	32.69	41.35	80.77	33.65	32.69	33.00
hepatitis	24.29	27.14	18.57	20.00	11.72	15.00
iris	5.00	5.00	12.50	7.50	2.50	5.00
labor	17.65	17.65	23.53	12.76	5.88	12.00
soybean	16.76	26.33	74.47	15.16	15.16	16.00
vote	5.09	6.94	14.35	6.94	2.59	5.00
water	60.28	38.65	57.80	41.48	38.65	39.00
waveform	45.84	31.45	34.01	38.76	31.45	32.00
wine	8.33	11.90	28.57	8.33	5.47	9.00
zoo	8.89	15.56	22.22	13.33	4.44	5.00
average	20.46	20.23	36.93	17.48	13.57	

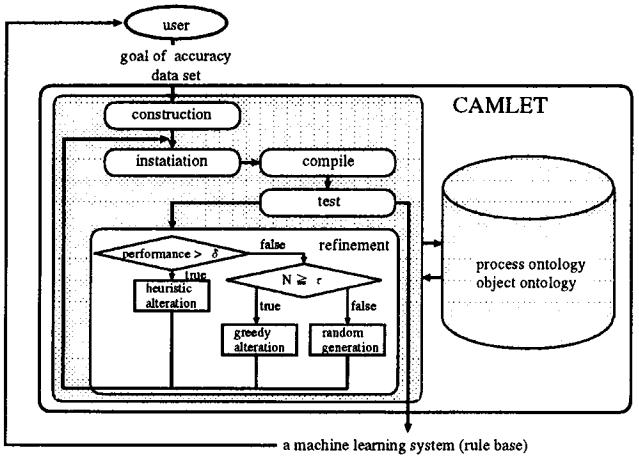

Figure 7: An Overview of CAMLET

CAMLET on UNIX platforms with C language, including the implementations of fifteen components in the process ontology with C language. We did case studies of constructing inductive learning systems for the fourteen different data sets from the UCI Machine Learning Repository. Five complete 5-fold cross-validations were carried out with each data set.

The results of these trials appear in Table 1. For each data set, the second to fifth column show mean error rates over the five cross-validations of popular inductive learning systems, such as C4.5 (J.R.,Quinlan, 1992), ID3, Classifier Systems and Bagged C4.5 (Leo Breiman, 1996). The sixth column contains similar results for inductive learning systems constructed by CAMLET. The final column shows error rates given as goal accuracy. Table 1 says CAMLET constructs inductive learning systems with best competence.

The systems constructed by CAMLET are specified in Table 2. Because the competence of the systems constructed by CAMLET are over those of popular inductive learning systems, it turns out for the grain size of process ontology to be proper for the task of composition of inductive learning systems. Table 2 shows us that CAMLET also in-

Table 2: Systems Constructed by CAMLET

Data	System				
	1	2	3	4	5
annealing	C4.5	C4.5	C4.5	C4.5(w)[1]	C4.5
audiology	B_C4.5	B_C4.5	B_C4.5	B_C4.5	B_C4.5
breast-w	AQ15	C4.5	New(1)[1]	AQ15	VS[1]
credit-s	ID3	C4.5	New(2)	New(3)	ID3
glass	C4.5	C4.5	C4.5	C4.5	C4.5
hepatitis	C4.5	New(4)	C4.5	New(3)	ID3
iris	C4.5(w)	C4.5(w)	C4.5(w)	C4.5(w)	ID3
labor	ID3	New(5)	New(5)	New(5)	C4.5
soybean	B_C4.5	C4.5(w)	B_C4.5	B_C4.5	B_C4.5
vote	New(4)	B_VS	New(4)	C4.5	VS
water	B_ID3	B_C4.5	ID3	ID3	ID3
waveform	CS	ID3	ID3	ID3	ID3
wine	B_ID3	C4.5	C4.5(w)	ID3	ID3
zoo	New(5)	New(5)	New(5)	C4.5	New(5)

vents new systems different from popular systems.

Figure 8 specifies a new system constructed by CAMLET. This system consists of bootstrap, star algorithm, windowling, and apportionment of credit.

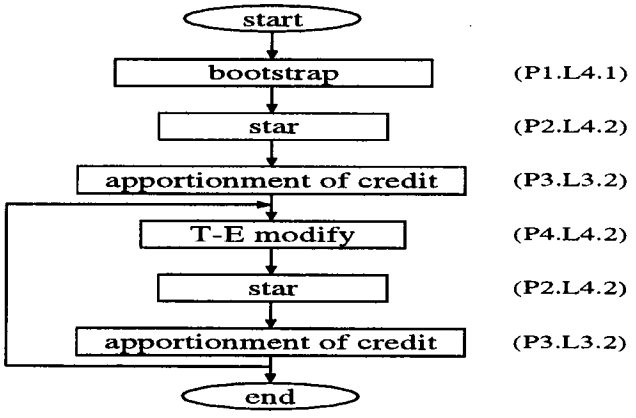

Figure 8: Specification of A New System – New (5) –

Table 3 shows the comparison of CAMLET and Neural Network from the point of mean error rates over five fold cross-validations. The competence of CAMLET is better than that of Neural Network on the average, except the following three data sets: annealing, breast-w and waveform.

Although the processes from Neural Network have been implemented in the process ontology, the case studies do not have them for constructing inductive learning systems. When a neural network is converted into another data type, such as classifiers, much information is missing. So we have just low performance of inductive learning systems constructed by CAMLET, using the processes from Neural Network. Maybe using the processes from Neural Network, we should take into account another meta-learning process without decomposing the whole control structure of Neural Network.

Table 3: Comparison of CAMLET and Neural Network

Data	NN	CAMLET	Data	NN	CAMLET
	err (%)	err (%)		err (%)	err (%)
annealing	1.00	9.00	labor	5.88	5.88
audiology	84.64	11.54	soybean	35.64	15.16
breast-w	4.42	5.36	vote	5.09	2.59
credit-s	22.24	13.53	water	43.97	38.65
glass	48.08	32.69	waveform	20.66	31.45
hepatitis	17.14	11.72	wine	17.86	5.47
iris	5.00	2.50	zoo	4.44	4.44
			average	22.57	13.57

[1] C4.5(w):C4.5 with windowling strategy, VS: Version Space (Michalski, 1983), New(x): new systems different from popular systems

Related Work

Although basic activities based on PSMs have usually been done by hands in constructing knowledge systems, CAMLET tries to automate the basic activities at the level of specifications (not yet at the level of codes).

Besides PSMs, there are several ontologies specific to processes, such as Gruninger's enterprise ontologies specific to business processes and PIF (Lee and Yost et al, 1994) and software process ontologies. Although our learning process ontology has some similarities to PSMs and other process ontologies about how to decompose processes, it decomposes processes, using more information specific in the field of task-domain (learning in our case).

From the field of inductive learning, CAMLET has some similarities to MSL (Raymond J.Mooney, 1994). MSL tries to put two or more machine learning systems together into an unified machine learning system with better competence. MSL does not decompose machine learning systems (the adaptation of machine learning systems sometimes comes up for interconnection). So the grain size of the components in CAMLET is much finer than the grain size of ones in MSL. Furthermore, MSL has no competence to invent a new machine learning system like CAMLET.

MLC++ (Kohavi.R, 1996) is a platform for constructing inductive learning systems. However, MLC++ has no facility for automatic composition of inductive learning systems like CAMLET.

Celine et al. also (Celine and Patrick, 1994) decompose learning systems and construct decomposition of bias similar to process ontology presented here. However, the decomposition of bias covers just limited inductive learning systems and automatic composition facility has not yet been done.

Conclusions and Future Work

This work comes from inter-discipline between machine learning and ontologies engineering. We put recent efforts on specifications and codes for ontologies and less on efficient search mechanisms to generate inductive learning systems with best performance. We need to make the refinement activity more intelligent and efficient with failure analysis and parallel processing. The refinement activity should move into an invention activity to invent new learning processes and new objects manipulated by learning processes.

References

Celine Rouveirol and Patrick Albert, 1994. Knowledge level model of a configurable Learning System. Springer-Verlag LNAI867, pp.374-393.

Gertjan van Heijst, 1995. The Role of Ontologies in Knowledge Engineering, Dr Thesis, University of Amsterdam.

Gruber,T.R., 1992. Ontolingua: A Mechanism to Support Portable Ontologies, Technical Report, KSL 91-166, Computer Science Department, Standford University, San Francisco, CA.

J.Breuker and W.Vande Velde, 1994. Common CADS Library for Expertise Modeling, IOS Press.

J.R.,Quinlan, 1986. Induction of Decision Tree, *Machine Learning*, 1, pp.81–106.

J.R.,Quinlan, 1992. C4.5 : Programs for Machine Learning, Morgan Kaufmann.

Kohavi,R. and Sommerfield, D, 1996. Data Mining using MLC++ - A Machine Learning Library in C++, Proc. 8th International Conference on Tools with Artificial Intelligence, pp.234-245.

L.B.Booker, D.E.Goldberg, J.H.Holland, 1989. Classifier Systems and Genetic Algorithms, *Artificial Intelligence 40*, pp.235-282.

Lee,J.G., Yost, and the PIF Working Group, 1994. The PIF Process Interchange Format and Framework, MIS CCS Working Report#180.

Leo Breiman, 1996. Bagging predictors, *Machine Learning*, Volume 24, Number 2, pp.123-140.

Michalski,R.S., Carbonell,J.G., Mitchell,J.M.(eds.), 1983. Machine learning: An artificial intelligence approach, Morgan Kaufmann, Los Altos, CA.

Musen M.A. et al., 1992. Overcoming the limitations of role-limiting methods, editorial special issue, Knowledge Acquisition, 4(1):162-168.

Raymond J.Mooney, Dirk Ourston, 1994. A Multistrategy Approach to Theory Refinement, *Ma-*

chine Learning: A Multistrategy Approach, Volume 4, pp.141-164.

T. Bylander and B. Chandrasekaran, 1987. Generic Tasks for Knowledge-Based Reasoning: The "Right" Level of Abstraction for Knowledge Acquisition, *IJMMS; International Journal of Man-Machine Studies*, Volume 26, pp.231-243.