# Multistrategy Task-adaptive Learning Using Dynamically Interlaced Hierarchies: A Methodology and Initial Implementation of INTERLACE

**Nabil W. Alkharouf and Ryszard S. Michalski***

Machine Learning and Inference Laboratory
George Mason University,
Fairfax, Virginia, 22030, USA
{nabil, michalski}@aic.gmu.edu

* Also with GMU Departments of Computer Science and Systems Engineering, and the Institute of Computer Science, Polish Academy of Sciences

## Abstract

This research concerns the development of a methodology for representing, planning and executing multitype inferences in a multistrategy task-adaptive learning system. These inferences, defined in the Inferential Theory of Learning as *knowledge transmutations*, are generic types of knowledge operators, and are assumed to underlie all learning processes. The paper shows how several basic knowledge transmutations can be seamlessly integrated using a knowledge representation based on *dynamic interlaced hierarchies* (DIH). The implemented system, INTERLACE, includes an interactive graphical user interface for visualizing knowledge transmutations that are being performed by the system. INTERLACE is illustrated by several examples.

## Introduction

The development of a multistrategy learning system capable of integrating different learning strategies according to the needs of a given learning task requires a knowledge representation that facilitates multitype inferences. The Inferential Theory of Learning (Michalski 1993), which provides a theoretical framework for multistrategy learning, views every learning process as a goal-oriented search through a knowledge space. In this search, the operators are instantiations of generic forms inference, called *knowledge transmutations*, the search goal is a specification/characterization of knowledge to be acquired, and the starting point is knowledge the system already possesses. A knowledge transmutation takes some input knowledge (e.g., a training example(s), a general statement, an observation, etc.), a relevant part of the learner's prior knowledge, and produces a new piece of knowledge (e.g., a generalization of examples, a consequence of the given statement, an explanation of the observation, etc.). Knowledge transmutations can be classified on the basis of the type of inference they employ and the type of knowledge they produce.

This paper describes a methodology and an initial implementation of the system INTERLACE, for supporting different knowledge transmutations and applying them according to the *learning task*. A learning task is determined on the basis of the input knowledge provided to the learner, the learner's prior knowledge, and a learning goal. The methodology employs a knowledge representation,, called *Dynamically Interlaced Hierarchies* (DIH), which builds upon research on human plausible reasoning (Collins & Michalski 1989), frame representation (Minsky 1975) and semantic networks (Quillian 1968). The DIH representation supports a seamless execution of a wide range of knowledge transmutations, and provides a backbone for implementing a Multistrategy Task-Adaptive Learning (MTL) system (Michalski 1993).

The DIH representation provides an expressive, modifiable and flexible knowledge representation system. The initial efforts in this direction were described in (Hieb & Michalski 1993). The DIH representation employs *part*, *type* and *precedence* hierarchies for representing hierarchical conceptual structures, and *traces* for representing statements involving concepts from different hierarchies.

INTERLACE combines the DIH representation with a control mechanism for executing knowledge transmutations in a seamless integrated manner, according to a given knowledge goal. A learning process is then a goal-directed transformation of knowledge that is carried out through inferential processes embodied in knowledge transmutation operators.

The INTERLACE system consists of three interacting modules, a knowledge base for DIH hierarchies, traces and transmutations, a goal-driven planner for generating plausible sequences of transmutations in order to achieve a given goal, and a graphical user interface for visualizing knowledge transmutations. The implemented system includes capabilities for such transmutations as inductive generalization, deductive specialization, abstraction and similization, and their counterparts, deductive generalization, inductive specialization, concretion and dissimilization. It also allows for the graphical creation, modification and deletion of DIH hierarchies and traces.

## Related Research

One of the important aspects of the Inferential Theory of Learning (ITL) is that provides an approach to analyzing and describing the *competence* of learning strategies, that is, their logical capabilities. Theory views every learning process as a search through a *knowledge space* defined by the employed knowledge representation, and guided by a learning goal. The search operators, instantiations of transmutations, represent different types of inference or knowledge transformation. Basic knowledge transmutations include generalization, abstraction, explanation, selection, association, similization, agglomeration, characterization, and their counterparts, specialization, concretion, prediction, generation,, disassociation, dissimilization, decomposition and discrimination (Michalski 1994).

According to ITL, to instill learning capabilities in a multipurpose intelligent agent, one needs to implement in it the ability to conduct multiple types of inference (in general, any type of knowledge derivation or transformation) and the ability to store and retrieve knowledge. This observation forms the basic premise of the theory, which can be succinctly stated in the following "equation":

*Learning = Inference + Memory*

This theory is conceptually close to Newell's ideas of about knowledge-level systems (Newell 1994). ITL abstracts from the medium that transforms knowledge, and concentrates on the type of knowledge transformations that occur in learning processes. Therefore, it can be used, in principle, for analyzing any type of learning. In this paper, ITL is used as a conceptual basis for implementing a *multistrategy task adaptive learning*.

Research on multistrategy task-adaptive learning (MTL) systems aims at synergistically integrating a wide range of learning strategies in order to perform different learning tasks. A learning task is defined by the available input to the learning system, learner's prior knowledge, nd a learning goal. The proposed system is called task-adaptive, because it is supposed to dynamically combine different learning strategies so that it can perform a given learning task (Michalski 1993, 1994; Michalski & Ram 1995). The system is inherently goal-dependent, because it does not assume that the learning process is just a search for a generalization of examples, as often done in machine learning, but can, in principle, search for any kind knowledge that is desirable by the learner (Michalski 1994).

Research on task-adaptive learning is conceptually related to research on integrated learning, (e.g., Bergadano, Giordana & Saitta 1991), and goal-driven learning (e.g., Hunter 1990; Leake & Ram 1995; Ram & Leake 1995). In general, multistrategy learning systems range from *loosely coupled* systems that consist of several learning modules coordinating to achieve the learning task to *tightly coupled* systems that deeply and synergistically integrate inferential learning strategies in order to achieve the desired learning performance. Examples of tightly ("deeply") integrated learning systems include the discussed learning system (MTL-DIH) and a related system based on plausible justification trees (Tecuci 1993).

Figure 1 presents a general schema of a multistrategy task-adaptive learning system based on DIH knowledge representation (a MTL-DIH system). The learning task is defined by the input DIH traces, relevant background knowledge in the form of DIH hierarchies and existing traces through the hierarchies, and a learning goal (in general, it can be a set of goals). A multitype inference engine utilizes DIH transmutations as knowledge generation operators under the supervision of a control and planning subsystem.

## Dynamically Interlaced Hierarchies

The idea of the knowledge representation based on Dynamically Interlaced Hierarchies stems from the observation that some knowledge is more or less stable and other knowledge is acquired incrementally using the stable knowledge as hooks. In (Collins & Michalski 1989), this "stable" knowledge is represented by type and part hierarchies. Initial ideas and a method for the implementation of Dynamically Interlaced Hierarchies (DIH) was presented in (Hieb & Michalski 1993). The DIH representation is conceptually related to semantic networks (Quillian 1968), but is has some important distinctive features, designed to facilitate a conceptual simplicy of representation and performing diverse forms of inference in a uniform way.

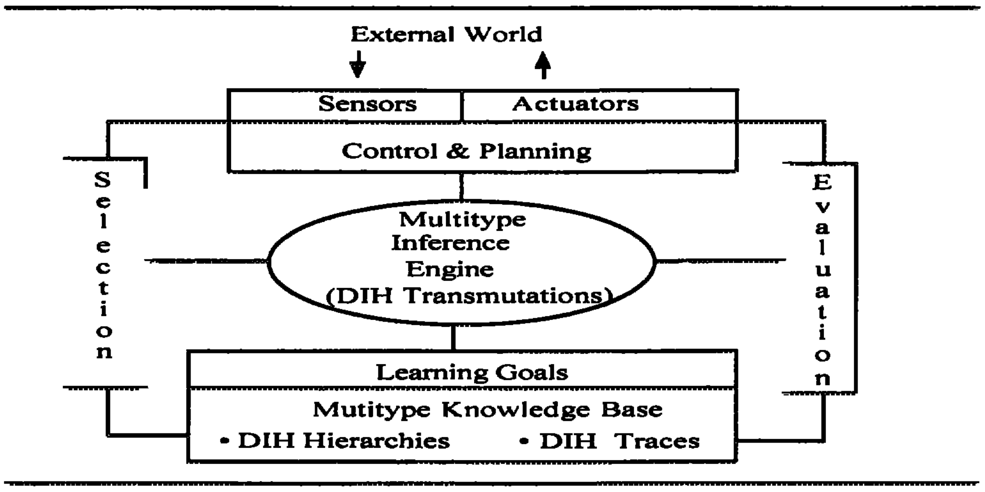


Figure 1. A general schema for a multistrategy task-adaptive learning system based on the DIH knowledge representation.

In DIH knowledge is partitioned into a *static* and *dynamic* part. The static part represents relatively stable knowledge organized into *concept hierarchies* such as part, type and precedence hierarchies, each hierarchy organizes its concept nodes from a certain context or viewpoint, with the possibility of concept nodes participating in more than one hierarchy. The dynamic part represents knowledge that changes relatively frequently. This knowledge is represented by *traces* that are paths linking nodes of different hierarchies in order to represent single statements.

DIH provides a suitable knowledge representation for facilitating multitype inference in multistrategy learning systems that require an expressive, modifiable and flexible representation language. Initial work on the DIH representation was presented in (Hieb & Michalski 1993).

## INTERLACE

INTERLACE is the core of the MTL-DIH multistrategy learning system. It consists of three interacting modules — a knowledge base, a goal driven planner and a graphical user interface. To implement INTERLACE, we needed a language that would facilitate the construction of concept hierarchies and making links among them. For this purpose we chose LIFE programming language, originally conceived at MCC.

LIFE is a synthesis of three different programming paradigms: logic programming, functional programming, and object-oriented programming. It is a declarative logic-based language that can be seen as a constraint language. It derives its syntax and resolution method from Prolog. It uses λ-terms as its basic data structures, and unification and matching as its basic operations.

LIFE semantically extends PROLOG in two ways:

1) Herbrand terms are replaced by λ-terms.

2) It includes *call by matching* (Prolog only supports call by unification).

As a programming language LIFE provides clean and elegant solutions to a number of Prolog's deficiencies. It includes functions (correct arithmetic), object-orientation, C-like records, expandable data-structures (arrays and hash-tables), types and multiple inheritance, correct manipulation of cyclic structures, coroutines and constraints, global variables, clean destructive assignment, persistent data structures.

## Representation of DIH Hierarchies and Traces

As mentioned earlier, DIH hierarchies represent relatively stable background knowledge. Such knowledge consists of concepts organized into hierarchies of different kinds, such as type, part and precedence hierarchies. A DIH node is represented in LIFE as a sort, an identifier standing for a set of objects. A sort corresponds intuitively to a type in a traditional programming language, or a class in an object-oriented programming language.

Sorts are organized into a hierarchy through a set of sort declarations (using the <| operation). For example, the navy_aircraft type hierarchy in Figure 2 is declared in the following manner:

```
combat_aircraft <| navy_aircraft
fuel_aircraft <| navy_aircraft
service_aircraft <| navy_aircraft
f16 <| combat_aircraft
tomcat <| combat_aircraft
```

DIH traces represent "dynamic" knowledge that changes relatively frequently. A trace roughly stands of a statement, and is a path connecting nodes of different hierarchies. DIH traces are represented using λ-terms, the LIFE's basic data structure. Each λ-term has the following syntax:

root_sort (label_1 => sort, label_2 => sort, ....)

Each sort including the root sort is a *part of* a hierarchy. Each sort can be further refined using the hierarchy that is also a *part of* type. This forms a basis for the functionality of DIH transmutations described below. For example, the DIH trace expressing the statement "Some aircraft carriers in Norfolk have fuel aircraft" is represented as:

aircraft_carrier(quantifier => some, location => norfolk, navy_aircraft => fuel),

where *aircraft_carrier* is the trace argument (root sort), *quantifier, location, and navy_aircraft* are viewed as descriptors applied to the trace argument. Adding knowledge to the DIH representation is done by creating hierarchies and specifying traces that express statements involving nodes of different hierarchies.

## Representation of DIH Transmutations

We will show now that several basic knowledge generation transmutations, as described in the Inferential Theory of Learning (Michalski 1991, 1992, 1993), can be easily performed in DIH, specifically, just by moving some links. The transmutations include generalization, abstraction, explanation and similization, and their counterparts, specialization, concretion, prediction and dissimilization, respectively. The DIH system currently implements six transmutations:

**QGEN:** Quantification-based Generalization
**QSPEC:** Quantification-based Specialization
**AGEN:** Argument-based Generalization
**ASPEC:** Argument-based Inductive Specialization
**ABST:** Abstraction
**CONC:** Concretion

By repeating these transmutations in different combinations, the system can perform a wide range of inferences. Every transmutation has the following syntax:

TransType(I_TRACE, CONTEXT, O_TRACE),

where I_trace is an input trace, and O_trace is an output trace (a trace generated by a transmutation).

An inference step performed by a transmutation can be depicted graphically as a movement of traces along some nodes of the hierarchies. To illustrate specific transmutations (operators) performed by INTERLACE, suppose that given is an input statement: "Some Norfolk aircraft carriers are fuel carriers."

This statement is represented by a trace:

I_TRACE: aircraft_carrier (quantifier => some, location => norfolk, navy_aircraft => fuel).

The order of arguments for the trace interpretation is defined by a schema hierarchy, SH, shown in Figure 2.

For the given input, the **QGEN** operator (Quantification-based Generalization) produces a trace:

O_TRACE: [aircraft_carrier (quantifier => most, location => norfolk, navy_aircraft => fuel)] (Figure 2)

Given the same input, the **QSPEC** operator (Quantification-based Specialization), produces:

O_TRACE = [aircraft_carrier (quantifier => one, location => norfolk, navy_aircraft => fuel)].

Again, given the same input, the **AGEN** transmutation (Argument-based Generalization) produces:

O_TRACE=[navy_battleship(quantifier=>some, location => united_states, navy_aircraft => fuel)] (Figure 2).

**ASPEC** (Argument-based specialization can produce one of several alternatives, depending on *merit parameters* (Collins & Michalski 1989). One of them (Figure 3):

O_TRACE = [iowa_class(quantifier => some, location => united_states, navy_aircraft => fuel)].

**ABST** (Abstraction) produces:

O_TRACE = [aircraft_carrier(quantifier => some, location => united_states, navy_aircraft => fuel)] (Figure 4).

**CONC** (Concretion), another form of inductive inference, produces (among other plausible alternatives):

O_TRACE = [aircraft_carrier(quantifier => some, location => united_states, navy_aircraft => b677) (Figure 5).

In principle, all the above transmutaions can represent deductive or inductive inference, depending on the combination of nodes in a trace and on what nodes are moved (Michalski 1994).

## Goal-Driven Learning based on DIH

The idea behind the DIH-based goal-driven learning is to apply a sequence of transmutations that transfer a given input (one or more traces) into one or more output traces that satisfy a learning goal. A learning goal is specification of knowledge that the learner wants to aquire. A learning goal is generated by a performance system, for example, as a request for some knowledge, as a result of an impass in problem solving (Newell 1994), etc. A goal specification is expressed as a trace. A goal trace may include high level nodes whose children are to be learned, or nodes indicating the type of knowledge to be acquired. Learning goals can be classified into two categories: domain specific and domain independent (Michalski & Ram 1995). Learning in INTERLACE system is designed to handle both domain specific and domain independent goals.

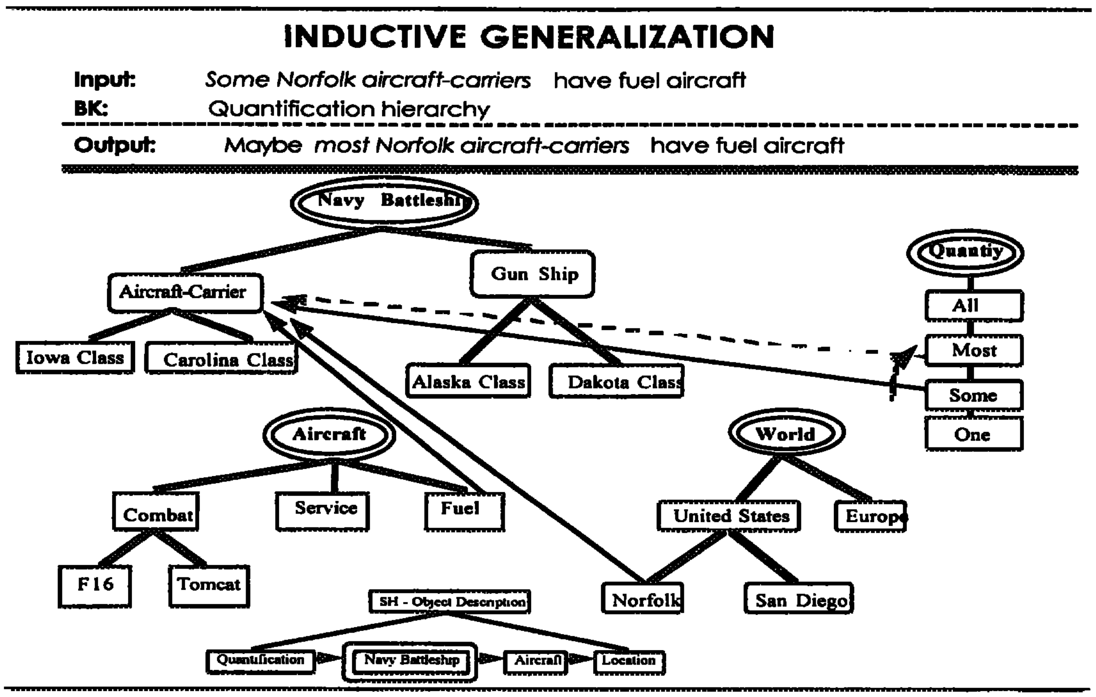


Figure 2. An example of a quantification-based inductive generalization.

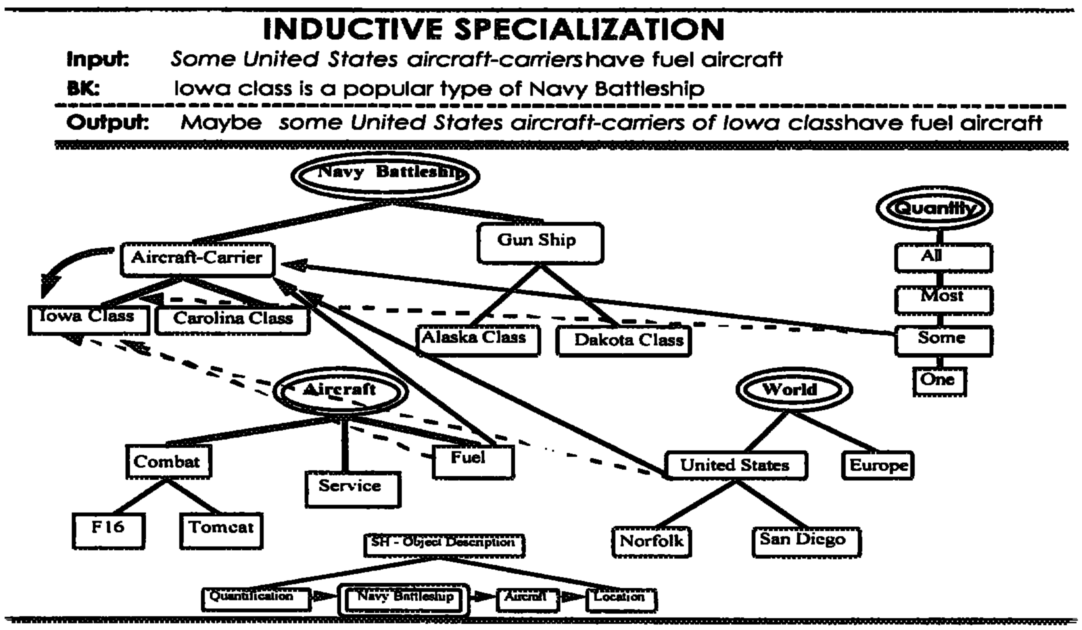
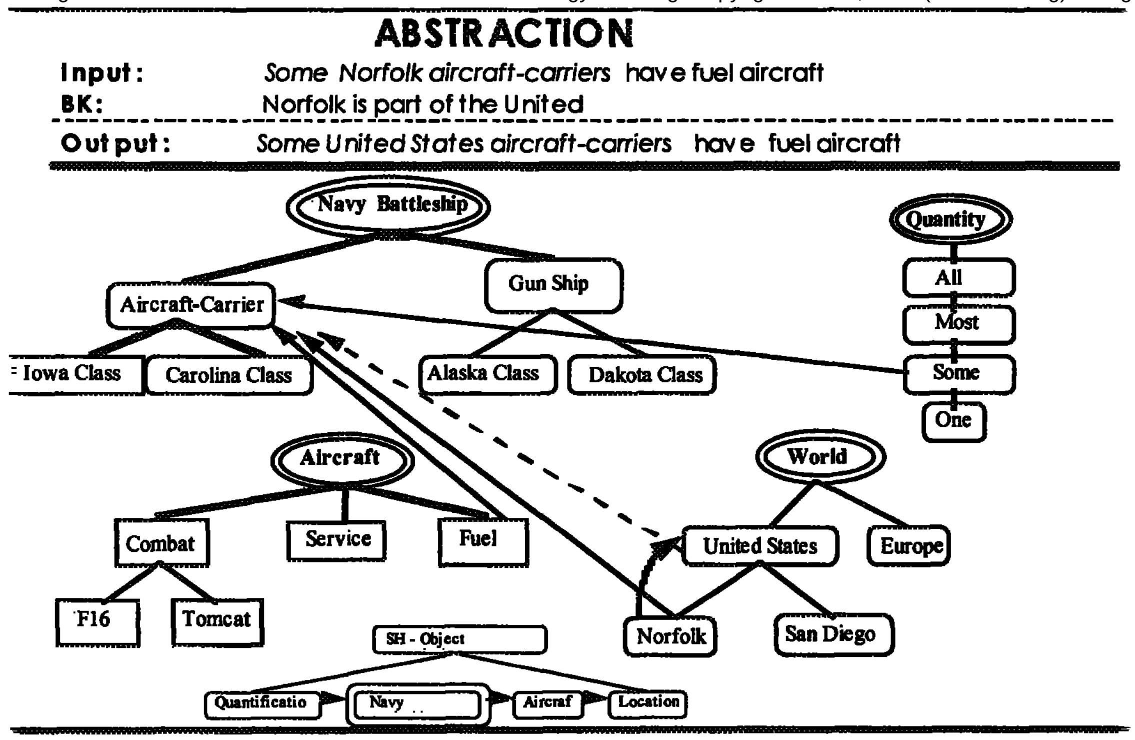


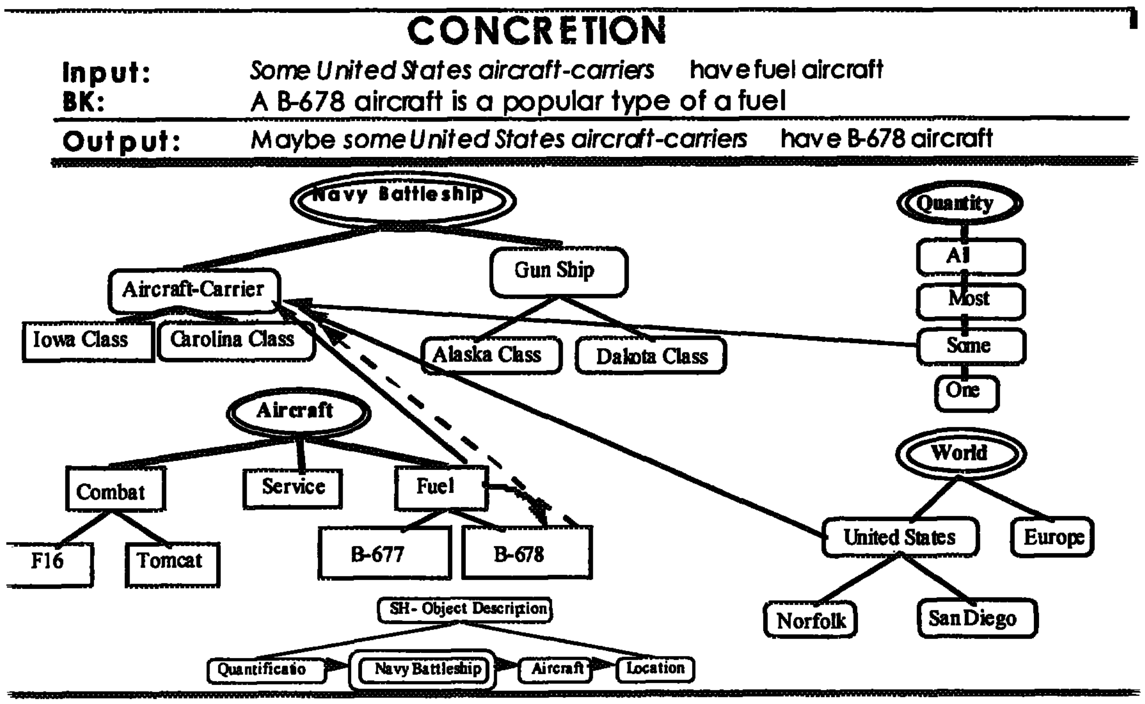
Figure 3. An example of an argument-based inductive specialization.



**ABSTRACTION**

**Input:** *Some Norfolk aircraft-carriers* have fuel aircraft
**BK:** Norfolk is part of the United
**Output:** *Some United States aircraft-carriers* have fuel aircraft

Figure 4. An example of abstraction transmutation.



**CONCRETION**

**Input:** *Some United States aircraft-carriers* have fuel aircraft
**BK:** A B-678 aircraft is a popular type of a fuel
**Output:** Maybe *some United States aircraft-carriers* have B-678 aircraft

Figure 5. An example of concretion transmutation.

**Domain Specific Goals.** Domain specific goals define knowledge that is relevant in the given domain. The task of the planner is to generate a sequence of transmutations that produce knowledge that satisifies the given goal.

Such a process is invoked in INTERLACE by:

Plan(I_TRACES, G_TRACES, CONTEXT, PLAN), where

I_TRACES — a list of relevant input traces

G_TRACES — a goal trace (or a list of traces).

CONTEXT — a context for the process, specified by a list of hierarchies within which the transmutations are to be performed)

PLAN — a list of plausible transmutations that proof the G_TRACES.

The planner determines which transmutation is appropriate for a given goal in the given contex Deductive transmutations are favored over inductive transmutations during the planning process.

To illustrate the process, let us consider: I_TRACES = [Iowa_class (quantifier => one, location => United_States, aircraft_type => tomcat) ], and G_TRACES = [aircraft_carrier (quantifier => most general, location => norfolk, aircraft_type => combat) ]. The planning process produces the following sequence of plausible transmutations to achieve the goal: argument-based deductive deneralization from Iowa_class to aircraft_carrier, abstraction from tomcat to combat, two inductive quantification-based generalizations — from one to some, and then to most, and an inductive concretion from United_states to Norfolk.

**Domain Independent Goals.** Domain independent goals require the system to determine some general type of knowledge, e.g., a generalization of examples, a hierarchical classification of given facts, an explanation for an observation, etc.). Such goals are handled by designated procedures. In the absence of a more specific goal, the system executes the universal learning goal — derive all plausible knowledge that can be learned from a given input. The universal goal is executed by analyzing the relationship between an input trace and background knowledge, and performing steps dependent on the result of this analysis, as illustrated in Figure 6 (Michalski 1994).

## INTERLACE Graphical Interface

An interactive graphical interface using OSF MOTIF (Johnson 1993; Young 1994) was utilized for the creation, modification and deletion of DIH hierarchies and traces, all actions performed by the user in the graphical interface are reflected in the underlying knowledge base. The interactive graphical interface has the following features:

**Hierarchy Creation.** Create a node anywhere on the screen. Two types of arcs are defined: Undirected and Directed. Each of the above arcs can be in solid line style or on-off dash style. Hierarchies are created by using undirected solid line style arcs. By a repeated application of "create an arc" between two nodes, one can create a hierarchy. The system allows a user to create multiple hierarchies within a screen.

**Trace Creation.** Traces are created by using directed on-off dash arcs connecting nodes from separate hierarchies. The tip of the directed on-off dash arc designates the trace argument. The tail of the directed on-off dash arc designates the referent.

**Hierarchy Movement and Orientation**

- Hierarchies can be viewed vertically and horizontally.
- Hierarchies can be centered on the screen
- Hierarchies can be selected and moved around on the screen
- A single node within a hierarchy can be moved around.
- Arcs can be moved to reflect new relationships between two nodes.
- Subhierarchies can be selected and moved around.

**Hierarchy and Trace Deletion**

- Delete a single selected node, this in turn deletes any arcs associated with the deleted nodes
- Delete a set of selected nodes, this in turn deletes any arcs associated with the deleted nodes
- Delete an arc
- Delete selected arcs
- Delete the whole hierarchy

## Conclusion

This paper described an ongoing research on the design and implementation of a task-adaptive multistrategy learning system based on the DIH knowledge representation. The DIH representation was shown to be very useful for performing a wide range of knowledge generation transmutations that are required in multistrategy task-adaptive learning.

Among the most important objectives for future research is how to represent different kinds of goals, and how to determine most appropriate transmutations in pursuing the given learning goal.

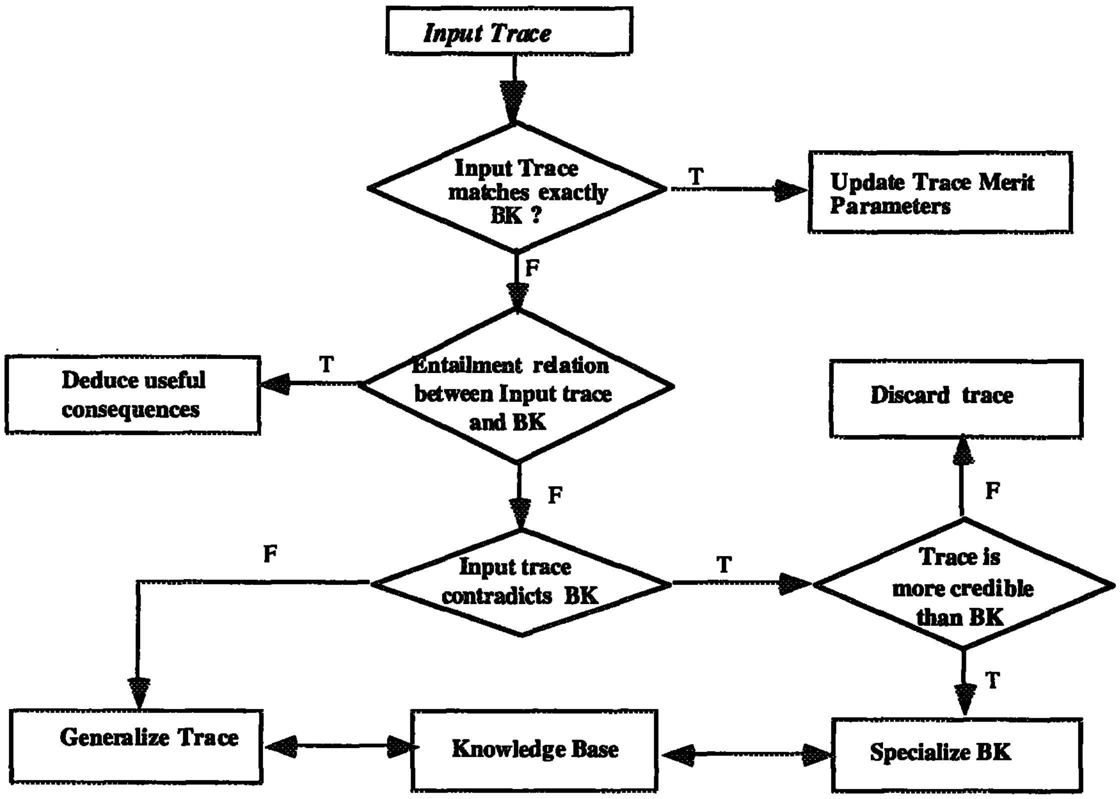

Input Trace
Input Trace matches exactly BK ?
T
Update Trace Merit Parameters
F
Entailment relation between Input trace and BK
T
Deduce useful consequences
Discard trace
F
F
Input trace contradicts BK
T
Trace is more credible than BK
F
T
Generalize Trace
Knowledge Base
Specialize BK

Figure 6. A simplified schema for multistrategy learning pursuing the universal learning goal.

## Acknowledgments

This research was done in the Machine Learning and Inference Laboratory at George Mason University. Authors thank members of the laboratory for valuable discussions and criticism.

The Laboratory's research is supported in part by the National Science Foundation under grants DMI-9496192 and IRI-9020266, in part by the Office of Naval Research under grant N00014-91-J-1351, in part by the Defense Defense Advanced Research Projects Agency under grant No. N00014-91-J-1854 administered by the Office of Naval Research, and in part by the Defense Advanced Research Projects Agency under grants F49620-92-J-0549 and F49620-95-1-0462, administered by the Air Force Office of Scientific Research.

## References

Bergadano, F., Giordana, A. and Saitta 1991. *Machine Learning: An integrated Framework and its Applications*, Ellis Horwood.

Collins A. and Michalski, R.S. 1989. Logic of Plausible Reasoning: A Core Theory, Cognitive Science, vol. 13.

Hieb, M., Michalski, R.S. 1993. Knowledge Representation for Multistrategy Task-Adaptive Learning: Dynamically Interlaced Hierarchies, In Michalski R.S. and Tecuci, G. (Eds.), *Proceedings of the Second International Workshop on Multistrategy Learning*, published and distributed by Machine Learning and Inference Laboratory, George Mason University, Fairfax, VA.

Johnson-Laird, P. 1983, *Mental Models*, Harvard University Press.

Johnson, E. F., Reichard, K. 1993. *Power Programming in Motif*, MIS Press, Second Edition.

Leake, D. and Ram, A. 1995. Learning, Goals, and Learning Goals: A Perspective on Goal-Driven Learning, Artificial Intelligence Review.

Michalski, R.S. 1993. Toward a Unified Theory of Learning: Multistrategy Task-adaptive Learning," in *Readings in Knowledge Acquisition and Learning: Automating the Construction and Improvement of Expert Systems*, B.G. Buchanan and D.C. Wilkins, Morgan Kaufmann, San Mateo.

Michalski, R.S. 1993. Inferential Theory of Learning: Developing Foundations for Multistrategy Learning, In R. S. Michalski and G. Tecuci (Eds.), *Machine Learning: A Multistrategy Approach* Morgan Kaufmann Publishers.

Michalski, R.S., Ram, A., 1995. Learning as Goal-Driven Inference. In A. Ram and D. Leake (Eds.). *Goal-Driven Learning*, Bradford Books/MIT Press, Cambridge, MA.

Minsky, M. 1975. A Framework for the representation of knowledge; In P. Winston (Ed.), *The Psychology of Computer Vision*, New York: MCGraw-Hill.

Newell, A. 1994. *Unified Theories of Cognition*, Harvard University Press.

Quillian, M.R., Semantic Memory, in Minsky (Ed.). *Semantic Information Processing*, Cambridge, MIT Press, 1968.

Popper, K., *Logic of Scientific Discovery*, New York, 1959.

Ram, A. and Leake, D. (Eds.) 1995. *Goal-Driven Learning*, Bradford Books/MIT Press, Cambridge, MA.

Tecuci, G., 1993. An Inferential Based Framework for Multistrategy Learning. *Machine Learning: A Multistrategy Approach*, Morgan Kaufmann Publishers.

Young, D., Pew, J., "The X Window System Programming and Applications with Xt", Prentice Hall, First Edition.