A Computational Framework for Granularity and its Application to Educational Diagnosis

Jim Greer and Gordon McCalla
ARIES Laboratory
Department of Computational Science
University of Saskatchewan
Saskatoon, CANADA S7N 0W0

Abstract

Many artificial intelligence systems implicitly use notions of granularity in reasoning, but there is very little research into granularity itself. An exception is the work of Hobbs [1985], which outlines several characteristics of granularity. In this paper we describe an approach to representing granularity which formalizes in computational terms most of Hobbs' notions, often refining and extending them. In particular two types of granularity have been delineated: aggregation and abstraction. Objects can be described at various grain sizes and connected together into a granularity hierarchy which allows focus shifts along either aggregation or abstraction dimensions. We briefly discuss how we have used granularity hierarchies in the recognition of novice LISP programming strategies and show how this enhances the recognition process and can lead toward planning appropriate feedback for the student.

1 Introduction

Our long term goal in this research is to show how the use of granularity can enhance the capabilities of intelligent tutoring systems. Granularity is an important part of instruction for two reasons. The first reason involves pedagogy. The level of generality or specificity at which a tutor chooses to present a topic, combined with the level of generality or specificity at which the student interprets the presentation, will affect the student's success at understanding instruction. Both tutor and student must be "on the same instructional wavelength". Shifting grain size in instruction must proceed smoothly, guided either by tutor or student.

The second reason is diagnosis. It is often difficult to precisely diagnose a student's problems. Students frequently exhibit bizarre or original behaviour which may be quite incomprehensible in detail. However, it is often possible to understand generally what a student is attempting to do. This knowledge can be used in designing appropriate feedback to the student and in focussing on points of ambiguity and misunderstanding. Thus in educational diagnosis, unlike other domains, being able to recognize student behaviour at coarse grain sizes is often useful.

In this paper we propose a model for granularity, and show how it can be used in the recognition of strategies that novice students use in solving recursive LISP programming problems. This recognition can occur at coarser or finer levels of granularity, corresponding to shallower or deeper understanding of student behaviour.

2 Background

Shifts in perspective from high level to low level and vice versa have been implicit in many AI systems, ranging from the level shifts in heuristic classification schemes (e.g. as discussed in [Clancey, 1985]), through the hierarchical reasoning used by various planning systems (e.g. ISaccrdoti, 1977]), the use of knowledge hierarchies to guide computer vision systems (e.g. [Mackworth and Havens, 1981]), and the representation of knowledge in semantic network schemes (e.g. [Levesque and Mylopoulos, 1979]). Such shifts can be interpreted as granularity shifts, but have seldom been viewed from this perspective.

An exception is the work of Hobbs [1985] which attempts to explicitly delineate the nature of granularity and to show how granularity can be used in representation and reasoning. Hobbs describes the following characteristics and properties of granularity:

• relevant predicate set (R) - Given a view of the world, ie. a particular situation of interest, only certain selected predicates from the global theory of the world are of interest. These are called "relevant predicates". These must be determined locally since they constitute the perspective from which the world is viewed in a particular situation.

• indistinguishability relation (-*) - Pairs of objects (interpreted broadly as objects, events, actions, agents, etc.) in the domain of interpretation are considered to be indistinguishable if and only if no relevant predicate can distinguish between them. Thus $x \sim y \Leftrightarrow (\forall p \in \Re)\ p(x) \equiv p(y)$.

• simplification mapping (K) - A detailed view of the world may be collapsed to a simpler view by means of a function K which maps the objects at one grain size to a simpler set of equivalence classes of objects at a coarser grain size, K also maps relevant predicates at the finer grain size onto new relevant predicates which make objects within the coarser-grained equivalence classes indistinguishable. Thus for some equivalence class C in the simpler theory, if K:V->C and K:W—>C then for all predicates K(p) in the simpler theory, v is indistinguishable from w.

• articulation - Articulation is the translation from a coarse-grained to a finer-grained theory. Relevant predicates

at the coarse-grained level arc decomposed into finer-grained predicates. Although Hobbs only talks in general terms about articulation, such decomposition would presumably be carried out using a mapping like K~1, which defines the classes of indistinguishable objects at finer grain sizes.

• idealization - Often the need to differentiate between objects at a coarse grain size forces the imposition of an arbitrary boundary between these objects, a process of idealization that is necessary to preserve the integrity of the coarse-grained classification. For example, if temperatures in the 60's form one such object and temperatures in the 70's form another, a predicate that could distinguish these two objects would need to be capable of distinguishing 69.9 from 70. This seems counterintuitive, given the grain size of the two classes being distinguished, but is seen by Hobbs as being preferred to fuzzy or probabilistic approaches.

The theoretical framework for granularity described by Hobbs has been the starting point for our investigations. We have been able to reinterpret Hobbs' notions in a computational formalism which both elaborates and refines the concept of granularity, as will be discussed below.

3 A Representation for Granularity

On closer analysis of granularity, it becomes apparent that there are at least two dimensions along which which granularity must be interpreted: abstraction, corresponding to shifts in focus from general to specific or vice versa; and aggregation, corresponding to shifts in focus through part-whole relationships. We propose a hierarchical representation for granularity in objects, roughly equating granularity with level shifts in a directed graph. Nodes in the graph are thought of as objects (broadly interpreted as in Hobbs), with links representing two distinct granularity relations, abstraction and aggregation.

Formally, a granularity hierarchy, y, consists of a finite set of objects, N, linked by granularity relations, i.e. the asymmetric binary relations $\overset{A}{\subset}: \aleph \rightarrow \aleph$ and $\overset{P}{\subset}: \aleph \rightarrow \aleph$ (corresponding to abstraction and aggregation respectively).

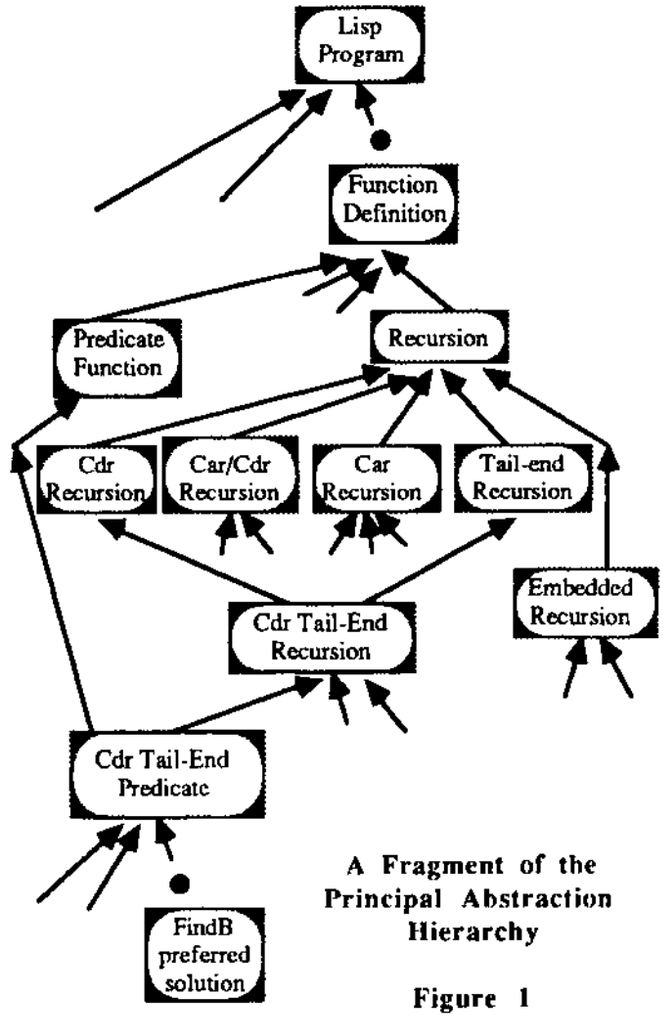
for $n_i, n_j \in \aleph \quad n_i \overset{A}{\subset} n_j \equiv \text{abstr}(n_j, n_i)$

which may be read n_j is an abstraction of n_i, or alternatively, n_i is a specialization of *n-,* and

for $n_i, n_j \in \aleph \quad n_i \overset{V}{\subset} n_j \equiv \text{aggreg}\ (n_i, n_i)$

which may be read n_j is an aggregation containing n_i or alternatively, n_i is a part of n_j. These two relations provide the links for a granularity hierarchy representing objects related by abstraction and/or aggregation. Objects may be maximal aggregations, which by definition are those objects which are not parts of any other objects, i.e. *n* is a maximal aggregation iff $\neg \exists n_i$ such that $n \overset{P}{\subset} n_i$.

A principal abstraction hierarchy, consisting of only these maximal aggregations, is a uniquely-rooted, directed acyclic graph with links corresponding to abstraction relations, $\overset{A}{\subset}$, connecting the maximal aggregation objects. The principal hierarchy represents the simplest (most aggregate) objects we wish to consider, arranged in terms of relative abstraction. This hierarchy is rooted at the most abstract object, and bottoms out at the most specialized objects. Figure 1 shows a fragment of a principal abstraction hierarchy for a set of LISP strategies at various grain sizes.



A Fragment of the Principal Abstraction Hierarchy

Figure 1

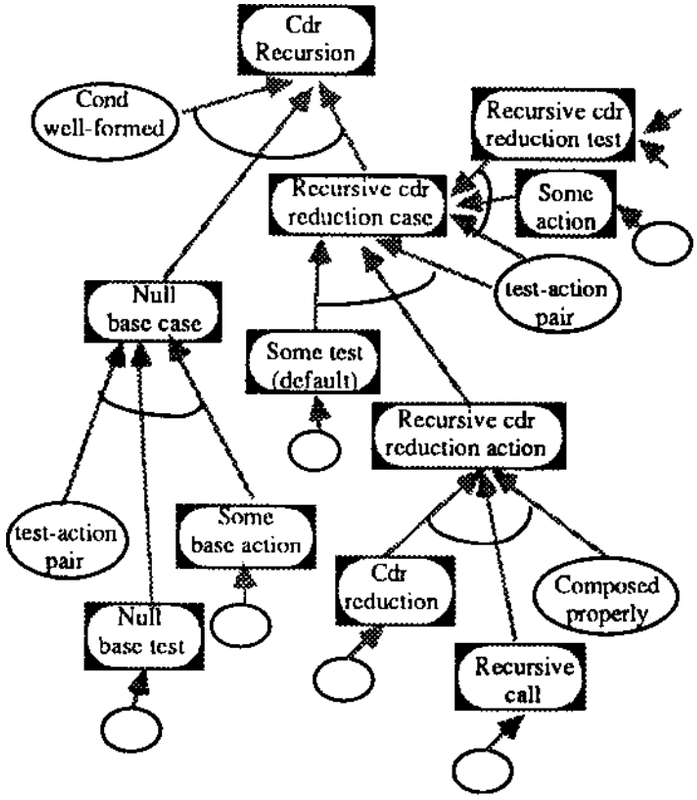
Each maximal aggregation object is the root of another directed acyclic graph, this time linked by aggregation relations p. Each object in this dimension is a component part of the maximal aggregation object, or a part of one of its parts, etc. Figure 2 shows the aggregation hierarchy rooted at the maximal aggregation "Cdr Recursion" object shown in the previous figure.

Abstraction and aggregation can be thought of as orthogonal dimensions of granularity, relating objects in the granularity hierarchy with one another. The entire granularity hierarchy is connected to the most general object, *(root),* in the principal abstraction hierarchy according to the connectivity axiom,

$$\forall n_i \in \aleph, (n_i \overset{A*}{\subset} root) \vee \exists n_j \in \aleph \mid (n_i \overset{P*}{\subset} n_j) \wedge (n_j \overset{A*}{\subset} root)$$

where $\overset{A*}{\subset}$ and $\overset{P*}{\subset}$ arc the transitive closures of $\overset{A}{\subset}$ and $\overset{P}{\subset}$ respectively. This implies that from any object the root can be accessed by shifting focus to more aggregate grain sizes until reaching a maximal aggregation, followed by shifting to more abstract grain sizes along the abstraction dimension. The root in the above figures is the object "Lisp program".

In addition to these linkages, we permit, but do not require, the existence of abstraction relations between finer-grained objects in the aggregation dimension. This provides for abstraction relationships among parts of finer aggregations. The resulting granularity hierarchy is a partial lattice of objects characterized by its two orthogonal relations. A more complete description of this formalization of granularity is given in another paper [Greer and McCalla, 1988].



A Fragment of the Aggregation Hierarchy for Cdr Recursion
Figure 2

Our approach to granularity can be interpreted in terms of Hobbs' characteristics, and in fact often refines and extends his notions. Each characteristic will be considered in turn:

• indistinguishability and distinguishability: Indistinguishability can be defined using the explicit structure of the hierarchy, y, rather than some set of relevant predicates. The indistinguishability relation is meaningful relative to a particular level of granularity (in abstraction and in aggregation). Objects are considered indistinguishable if they are finer grained than the object under consideration. Hence, there is a notion of indistinguishability with respect to each and every object in the hierarchy (denoted as $\sim_n$). We define indistinguishability between objects n\ and n_j with respect to an object n as

$$n_i \sim_n n_j \equiv ((n_i \overset{P*}{\subset} n) \wedge (n_j \overset{P*}{\subset} n)) \vee ((n_i \overset{A*}{\subset} n) \wedge (n_j \overset{A*}{\subset} n))$$

which indicates that descendants of an object, in either the abstraction or aggregation dimension are indistinguishable from the point of view of that object.

A related characteristic, not directly discussed by Hobbs, is distinguishability. Intuitively, one may think that this is simply the dual of indistinguishability, but it is not. Objects which can be distinguished with respect to a given object are precisely the other siblings of the object in both the aggregation and abstraction dimension. This relation gives an object knowledge of the perspective it embodies relative to local alternative perspectives. Clearly some objects may be neither indistinguishable nor distinguishable relative to some given object. Such objects are simply irrelevant to the given object, at least with respect to granularity considerations.

Looking at figure 1, from the point of view of "Cdr Recursion", "Car/Cdr Recursion", "Car Recursion", "Tail-end Recursion", and "Embedded Recursion" can all be distinguished from one another (being siblings). Further, "Cdr Tail-End Recursion", "Cdr Tail-End Predicate", etc. down to "FindB preferred solution" are all indistinguishable; and "Predicate Function", "Recursion", "Function Definition", and "Lisp Program" are undefined.

• relevant predicates: We refine Hobbs' notion of relevance to incorporate two kinds of predicates: observations and K-clustcrs. These not only define the aggregation level of granularity of an object, but arc utilized in object recognition at a particular level of abstraction granularity, as will be discussed in section 4.

An observation is a predicate completely determined by evidence obtained directly from the environment through the evaluation of some observer function (ofunction). An observation may be associated with an object for the purposes of distinguishing it from other objects, or to contribute to the recognition of the object on the basis of outside evidence. In fact, at some grain size(s) objects must be recognizable from direct observations alone. Thus aggregation "bottoms out", and every finest grained object along the aggregation dimension ultimately must be recognized by a direct observation.

Frequently an object can be characterised in a number of ways, even at a specific grain size. It may be characterized in terms of its parts and predicates that describe how those parts fit together; it may be characterised by some subset or other set of parts under certain conditions; or it may be characterized by predicates that are quite independent of its parts. For example, a recursive LISP function may be characterized by 1) a combination of recursive function parts (base cases and recursive cases) properly assembled, 2) an infinite tail-end recursion (as in an interpeter), or 3) particular behaviour in traces of function calls and returns. We name each different way of characterizing an object a "K-cluster" [Nilsson, 1981]. A K-cluster is a combination of observations and component parts that characterize an object under certain conditions. Thus, K-clusters occur along the aggregation dimension of the granularity hierarchy. The K-cluster provides a mechanism for effectively grouping relevant predicates into relevance groups.

In figure 2 there are a number of K-clusters, indicated by the arc connecting sub-aggregate descendants beneath a node. Observers in the figure are denoted as circles rather than boxes. For example, the object "Recursive cdr reduction case" has two K-clusters, one consisting of the two sub-aggregate objects "Some test (default)" and

"Recursive cdr-reduction action" as well as the observation "test-action pair", and the other consisting of the two sub-aggregate objects "Recursive cdr-reduction test" and "Some action" as well as the observation "test-action pair". Each of these K-clusters represents a different way in which a recursive cdr reduction case could be framed in LISP, and hence define alternative groups of relevant "predicates" which must be "true" for the object "Recursive cdr reduction case" to be distinguished. The truth or falsity of such predicate groups is determined by a gestalt function described below.

• simplification and articulation: Simplification in granularity hierarchies is accomplished by a focus shift from a particular level of granularity to a coarser grain size. A simplification operator (similar to Hobbs' K mapping) is required to guide these focus shifts. In the abstraction dimension, simplification amounts to traversing an abstraction relation, which implicitly alters the sets of distinguishable and indistinguishable objects.

In the aggregation dimension, the presence of K-clusters impacts the simplification process. Each object that could be a candidate for aggregation simplification (objects that are not maximal aggregations) is by definition a member of one or more K-clustcrs. Associated with each K-cluster is a function, called a gestalt function, which arbitrates the simplification and articulation of K-clusters. The gestalt function is essentially a local interpreter for the K-cluster which determines whether the objects (parts and observations) in the K-cluster are put together correctly. For example, in figure 2, the gestalt function associated with the K-cluster of parts of the "Cdr Recursion" object will make sure that the null base case and recursive cdr reduction case are in fact embedded in the observed cond function in an appropriate manner (i.e. the null case must precede the reduction case in the cond) before it will allow the "Cdr Recursion" object to be distinguished. The gestalt function has proven to be particularly useful for guiding aggregation focus shifts when granularity hierarchies are employed for recognition (as described in section 4).

Articulation is accomplished by a focus shift from one level of abstraction granularity to a finer grain size. This type of focus shift is the inverse of simplification, but has quite different semantics when applied to recognition tasks. Again the gestalt function is employed to arbitrate articulation between an object and one of its parts in a K-cluster.

• idealization: The need to impose fine-grained distinctions or boundaries between coarser-grained objects poses a difficult problem, only partially resolved in our framework. In contrast to Hobbs' framework where predicates relevant to coarser-grained objects are themselves necessarily coarser-grained, we permit but do not impose such restrictions. For example, in figure 2, "Null base case" (one of the relevant objects necessary to distinguish "Cdr Recursion") could have specializations or generalizations at finer or coarser levels of abstraction. These specializations or generalizations could, in turn, be objects relevant to distinguishing descendants or ancestors of "Cdr Recursion" in the principal hierarchy. However, such correspondences do not need to exist. This allows us to incorporate specialized relevant predicates at any level of abstraction and solves the fine-grained boundary problem in a manner similar to Hobbs' idealization approach.

On the other hand, this solution prevents the delineation of planes across the hierarchy at a particular level of abstraction (except in the special case where a complete lattice is defined by abstraction relations on aggregate parts). Such planes would roughly form a complete theory of the world at some abstraction grain size. In the recognition system that we have built using this theoretical framework, there has been no need for such a complete theory of the world at a level of abstraction, and in fact there are advantages to being able to connect objects to each other without regard for maintaining such planes (for example we have been able to avoid the need for plane-preserving intermediate dummy nodes required in Mulder's scene recognition vision system [Mulder, 1985]).

4 Using Granularity in LISP Program Strategy Recognition

Our main interest in using granularity has been to recognize the strategics novice students employ when they solve simple recursive LISP programming problems. This work has been carried on in the context of the SCENT project [McCalla and Greer, 1988], which is investigating issues in the construction of an intelligent tutoring system that dispenses advice to students as they learn to program in LISP. Using granularity, we are able to recognize student programming strategies at various levels of detail, which can be useful pedagogically and can enhance the robustness of the diagnostic system by allowing at least coarse grained recognition of bizarre student solutions. In fact, in the granularity hierarchy that we have implemented for LISP programming, strategy recognition at some grain size is guaranteed, albeit sometimes at only a coarse grain size.

To give a flavour for how the granularity-based recognition system works, consider the following simple LISP program, which a student might submit as his or her solution to the FindB problem (the FindB problem is to return T if the atom B exists at the top-level of a list, and to return NIL otherwise):

```
(defun FindB (Lst)
   (cond ((null Lst) nil)
         ((atom 'B) t)
         (t (cons nil (FindB (cdr Lst))))))
```

This is a flawed solution, in that the task-specific test "(atom 'B)" does not check for a B in the list, and there is no need for the "cons" composition after the recursive call. Most program recognition systems would find such a perturbed solution hard to deal with unless these particular perturbations had been explicitly anticipated. In our granularity-based approach, this program could at least be recognized as "Cdr Recursion" (see Figure 2).

In order to be a cdr recursion, a program must have a null base case, a recursive cdr reduction case, and these must be put together in a well-formed conditional. A null base case, in turn, must have a null base test and some base action, both put together as a test-action pair. Observers looking at the FindB solution above would find that "(null Lst)" is a satisfactory null base test, that "nil" is a suitable base action, and that the two are formed as a test-action pair. Thus, this program has a null base case. Does it have a

recursive cdr reduction case? There are two ways of having a recursive cdr reduction case, as shown by the two K-clusters in figure 2. The relevant K-cluster here is the one requiring some test (possibly a default) and a recursive cdr reduction action combined as a test-action pair. The student's use of "t" in the third clause of the cond can be recognized as a default test, but recognizing a recursive cdr reduction action involves a further aggregation articulation requiring a cdr reduction and a recursive call, properly composed. Observers can recognize "(cdr Lst)" as a cdr reduction, the call to "FindB" as a recursive call, and that these are composed properly (i.e. that the reduction is in the argument to the recursive call). Thus, a recursive cdr reduction action can be recognized. It remains only to observe that the default test "t" and this recursive cdr reduction action form a test-action pair, which means that a recursive cdr reduction case is recognized. The recognition of both the null base case and the recursive cdr reduction case, combined with the observation that the cond is well-formed, means that a cdr reduction has occurred. Since "Cdr Reduction" is an object in the principal hierarchy, further aggregation simplification cannot occur. Despite its perturbations, the student's program has been definitely recognized as a cdr recursion.

The recognition process proceeds as recognition of "Cdr Recursion" automatically propagates upwards through levels of abstraction simplification, allowing "Recursion", "Function Definition", and "Lisp Program" all to be recognized as well (see figure 1). In fact, once an object has been recognized at any level of aggregation, recognition propagates upwards through its abstraction ancestors, a fact which often allows rapid recognition of coarser grained sub-aggregate objects without needing to articulate all of their component parts. Using this information, the SCENT system knows what the student is doing at any of 4 levels of abstraction granularity, as well as being able to understand more precisely the various parts of the student's program that have been recognized in the aggregation dimension. This can be useful for updating the student model, and for framing the discussion with the student by focussing on the parts of the program that are correct and well understood.

This degree of recognition is insufficient, however. Pedagogically, it is the unrecognized parts of the student's program which usually form the basis for tutoring. The granularity-based recognition system is also quite useful in this regard. In the attempt to recognize the student's program, various objects elsewhere in the granularity hierarchy may have been recognized. These can include sub-aggregate objects of finer grained objects in the principal abstraction hierarchy. In particular, objects like "Cdr Tail-End Recursion", "Cdr Tail-End Predicate", and "FindB preferred solution" may all have some of their sub-aggregates recognized, while obviously not being able to recognize a complete K-clustcr of objects. For example, "FindB preferred solution", which is the finest grained object along the abstraction dimension, would be satisfied with the null base test, but would not accept the task-specific test (should be "(eq(car Lst)'B)"), nor would it be satisfied with the existence of a composition step. These unfelicitous parts could form the basis for tutoring, or at least for inquiries of the student or the student model as to problem-solving intentions. Thus, the two kinds of recognition, complete coarse grained recognition and finer grained partial recognition, provide the rest of the tutoring system with different sorts of relevant information for student modelling and pedagogical decision making.

The approach to granularity-based recognition discussed here has points of similarity to several other major research projects. In the domain of intelligent tutoring, the PROUST system [Johnson and Soloway, 1984] recognizes student programs at three levels: problem, goal, and plan. Our strategies are most like PROUST's plans, but unlike PROUST, we do not attempt to induce a student's goals in choosing a particular strategy (this is a role envisaged in SCENT for the student modelling component), nor do we formalize the problem description (although our work is currently progressing in this area). Instead, our approach has concentrated on strategy (plan) diagnosis, and in that regard goes considerably beyond PROUST in the formalization and use of granularity, in the delineation of both an aggregation and abstraction dimension to strategy recognition, in robustness, and in breadth and depth of strategies dealt with.

Knowledge-based vision systems, such as the various Mapsee systems [Mackworth and Havens, 1981, Mulder, 1985] and ALVEN and CAA [Tsostsos and Shibahara, 1987], have strong points of similarity with our approach to recognition as well. These systems, which use hierarchies of visual knowledge to guide recognition of scenes, bear similarity in organization of knowledge into aggregation and abstraction hierarchies to guide recognition. There are, of course, obvious differences in domain: the Mapsee systems recognize an idealized sketch map, ALVEN looks at medical images, and CAA analyzes electrocardiograms. None of these systems explicitly formulate recognition in terms of granularity, nor are they satisfied, in general, with only achieving coarse-grained recognition. The usefulness in an education domain of coarse grained recognition and partial fine grained recognition does not carry over very well to scene analysis. There arc also many technical differences between our approach and the knowledge-based vision approaches. Nevertheless, the important point is that the knowledge-based vision systems provide further evidence that the approach taken here may be widely applicable beyond program strategy recognition.

5 Conclusion

Much work has gone into the creation of a granularity-based recognition system for LISP strategies, and this work continues. One line of development has been to investigate various kinds of control paradigms. We have experimented with top-down, bottom-up, and task-dependent control schemes, and work by Barrie [1988] has investigated how to use so-called "strong" and "weak" recognizers to help direct the search through the granularity hierarchy. In fact, Barrie's initial work some years ago on strategy recognition launched our investigations into the use of granularity in recognition. Another line of development, explored by Pospisil [1988], has been to incorporate buggy strategies into the hierarchy, which if recognized allow a definitive diagnosis of the student's misconceptions, and provide even more concrete information for the student modelling and pedagogic components of SCENT. A third direction of current

investigations has been the knowledge engineering of a large system in order to rigorously test this approach to recognition, to prove out its usefulness in a real domain, and to find limitations and/or to explore possible enhancements to the approach. The design of the strategy objects in this large system is based on a repository of actual solutions to several LISP problems collected from some 48 novice LISP programmers (sec [Escott and McCalla, 1988] for an analysis of this data). We currently have implemented some 200 objects connected together at ten levels of abstraction granularity and averaging approximately four levels of aggregation granularity. These objects allow the recognition of a wide variety of the basic recursive strategies used by LISP programmers. We are enhancing the coverage of the hierarchy by adding more strategy objects and by integrating into the system knowledge-based program transformations in order to reduce the observers' dependency on exact code matches [Huang and McCalla, 1988]. We are currently attempting to reduce the need to store explicit task-dependent strategies at the finer levels of abstraction granularity and instead to generate these task-dependent strategies from the problem description as new tasks arc created for students to solve. Finally, we would like to find other domains where granularity-based recognition would be useful; some possibilities which we have considered include recognizing strategies used in software testing and debugging, and recognizing strategics employed by chess players.

Much work has gone into the formalization of granularity as well. We have been able to describe granularity in precise computational terms, have characterized two kinds of granularity, and have shown how this approach to granularity relates to Hobbs' properties of granularity, and in some ways refines and extends his ideas. We would like to explore our approach to granularity further, especially to define a notion of relative granularity between coarser and finer grained objects, to investigate how groups of objects at a similar relative grain size can be put together into a coherent "theory" of the world at a particular grain size, and to look at the implications of these variously grain-sized theories for representation and reasoning. Another interesting avenue to explore is the delineation of other kinds of granularity besides aggregation and abstraction. In particular, the human ability to carry out goals and sub-goals may suggest the existence of a dimension of granularity involving focus shifts between levels of goals. Perhaps other such dimensions exist as well. We believe that explicitly investigating granularity will prove to be useful for artificial intelligence, both theoretically and practically, and we are optimistic about our ongoing research into the formalization and use of granularity.

Acknowledgements

We would like to thank Bryce Barrie and Paul Pospisil for helping recognize the need for granularity-based representations. We would also like to thank Dan Baril, Shawkat Bhuiyan, Xueming Huang, and Mary Mark for their help in current investigations into granularity. The financial support of the Natural Sciences and Engineering Research Council of Canada is gratefully acknowledged.

References

[Barrie, 1988] Barrie, J.B. Using Granularity Hierarchies for Strategy Recognition, M.Sc. Thesis, Dept. of Computational Science, U. of Sask, Saskatoon, Canada, 1988.

[Clancey, 1985] Clancey, W.J. Heuristic classification, Artificial Intelligence, 27:289-350,1985.

[Escott and McCalla, 1988] Escott, J.A. and McCalla, G.I. Problem solving by analogy: a source of errors in novice programming, pages 312-319, Proc. International Conference on Intelligent Tutoring Systems, Montreal, Canada, June, 1988.

[Greer and McCalla, 1988] Greer, J.E. and McCalla, G I. Formalizing granularity for use in recognition, Applied Mathematics Utters, 1(4), 347-350,1988.

[Hobbs, 1985] Hobbs, J.R. Granularity, Ninth International Joint Conference on Artificial Intelligence, pages 432-435, Los Angeles, California, August, 1985.

[Huang and McCalla, 1988] Huang, X. and McCalla, G.I. A hybrid approach to finding language errors and program equivalence in an automated advisor, Proc. 7th National Conference of the Canadian Society for Computational Studies of Intelligence (CSCSI/SCEIO), pages 161-168, Edmonton, Canada, June, 1988.

[Johnson and Soloway, 1984] Johnson, W.L. and Soloway, E. Intention-based diagnosis of programming errors, Proc. 5th Conference of American Association for Artificial Intelligence (AAAI), pp. 162-168, Austin, Texas, August, 1984.

[Levesque and Mylopoulos, 1979] Levcsque, H. J. and Mylopoulos, J. A procedural semantics for semantic networks", in N. V. Findler (ed) Associative Networks, Academic Press, New York, 1979, 93-120.

[Mackworth and Havens, 1981] Mackworth, A.K. and Havens, W.S. Structuring domain knowledge for visual perception, Proc. 7lh International Joint Conference on Artificial Intelligence, pages 625-627, Vancouver, Canada, August, 1981.

[McCalla and Greer, 1988] McCalla, G., Greer, J., & the SCENT Research Team, Intelligent advising in problem-solving domains: the SCENT-3 architecture, pages 124-131, Proc. International Conference on Intelligent Tutoring Systems, Montreal, Canada, June, 1988.

[Mulder, 1985] Mulder, J. A. Using discrimination graphs to represent visual knowledge, Ph.D. Thesis, TR 85-14, Dept. of Computer Science, U. British Columbia, Vancouver, Canada, 1985.

[Nilsson, 1981] Nilsson, N.J. Principles of Artificial Intelligence, Tioga, Menlo Park, 1981.

[Pospisil, 1988] Pospisil, P.R. Diagnosising strategy errors in SCENT, M.Sc. Thesis, Dept. of Computational Science, U. of Sask, Saskatoon, Canada, 1988.

[Sacerdoti, 1977] Sacerdoti, E.D. A Structure for Plans and Behavior, Elsevier, New York, 1977.

[Tsostsos and Shibahara, 1987] Tsostsos, J. and Shibahara, T. Knowledge organization and its role in temporal and causal signal understanding: the ALVEN and CAA projects, page 221-261, in Cercone & McCalla (eds), The Knowledge Frontier, Springer-Verlag, 1987.