VICTORIA UNIVERSITY OF WELLINGTON

Department of Computer Science

PO Box 600
Wellington
New Zealand

Tel: +64 4 471 5328
Fax: +64 4 495 5232
Internet: Tech.Reports@comp.vuw.ac.nz

Representing and Matching Physical Objects

David Andreae

Technical Report CS-TR-93/1
February 1993

Abstract

We are developing a system called GRAM that can learn structural concepts from examples in the domain of two-dimensional physical objects. The components of the system are intended to be eventually incorporated into a general purpose instructable robot. GRAM is given information from a vision system about an observed object, from which it constructs a hierarchically organised part-based description. This is matched with an existing concept description, and the resulting correspondences between the parts are used to generalise the concept appropriately.

This paper gives a brief overview of the representation scheme and the matching algorithm.

Publishing Information

This report is to be published in the NZ Journal of Computing special issue, 1993.

Author Information

David Andreae is completing a PhD in Computer Science at VUW.

1. Introduction

We are developing a system called GRAM that can learn generalised descriptions of classes of structured physical objects (such as chairs, distributor caps, bedrooms, and bookshelves). The system is intended to be eventually incorporated into a general purpose instructable robot, such as a household robot that can learn to navigate around a house, vacuum floors, tidy rooms, and so on, or a warehouse robot that must arrange stock and notice when items are broken or misassembled.

GRAM learns a concept (such as 'chair') by being given descriptions of instances of the concept, one at a time, each of which is used to generalise the existing concept. GRAM can then use its concept descriptions to classify unknown objects, and to identify any unusual or unexpected features of an observed instance. For example, a description of a class of bicycle might be used to recognise a new bicycle that arrives at a warehouse, and to notice that the rear mudguard is shifted round from its usual position.

GRAM consists of a representation scheme, an instance-constructor, a matcher, and a generaliser. A vision system is assumed to be available to provide a description of an observed object or scene at multiple levels of detail. The instance-constructor augments the description in various ways to support matching and generalisation, such as identifying the important relationships between components, and finding groups of similar components that can be clustered together as a single entity. The matcher then finds correspondences between the components of the object description and the components of the concept description which is to be generalised. The generaliser uses these correspondences to generalise the concept appropriately.

Currently we deal only with two dimensional representations of objects, such as the humanoids in Figure 1, but the representation has been designed with the intention of extending to three dimensions. Concepts are described only in terms of physical *structure*, rather than behaviour or function.

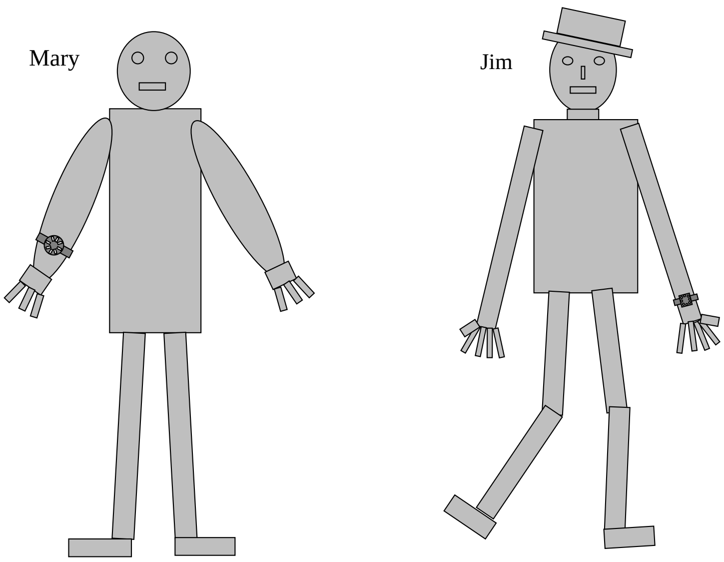

Figure 1: Two Humanoids

The contributions of the thesis are primarily the representation scheme, the mechanisms for constructing instance descriptions, and the methods of matching and generalising descriptions in a flexible and robust manner. Other systems such as (Thompson and Langley, 1991; Winston, 1975; Dietterich and Michalski, 1981; Connell and Brady, 1987; Wasserman, 1985; Brooks, 1981) either had an insufficiently rich representation scheme to describe the object-categories representable in GRAM, or were not able to match and generalise such objects effectively because of their rigid search strategies. This paper briefly describes some of the features of GRAM's representation, and also gives an outline of the matching algorithm.

2. The Representation

Several principals and requirements have guided the design of the representation, and these include the following:

We wanted the representation to be sufficiently rich to be able to capture information necessary to, firstly, recognise instances of concepts such as household objects, and, secondly, to identify important differences (or 'faults') between an observed object and a concept. GRAM does not deal with functional knowledge about a concept, and so we wanted the structural descriptions to capture much of the information that would be implicit in functional specifications. For example, a functional description of a chair might state that a chair must be 'stable' and that it is used for people to sit in. A structural description should, therefore, at least be able to specify that the lengths of the legs of a chair are of the same length, and that its context may include a person in a particular posture.

We also wanted the representation to include multiple levels of abstraction and approximation, as discussed by (Marr, 1982). This not only guides the matcher, but also enables objects to be compared and generalised at a coarse level of detail, even when their finer details differ significantly.

We required descriptions of concepts to be probabalistic. In other words, the frequencies of its features should be explicit, thus enabling the system to determine which features are most important or most likely to be present in an observed instance.

We wanted to be able to explicitly represent collections or sequences of similar components, such as the buttons on a telephone, the drawers of a desk, or the petals of a flower. These groups should be representable as a single entity that can be matched and generalised as a whole, thus avoiding the need to represent, match, and generalise every component individually.

We also wanted to be able to represent concepts for which the substructure could have a variety of arrangements, such as the concepts 'bedrooom' and 'desktop', rather than the relatively constrained substructure arrangements for concepts such as 'saucepan' and 'egg-beater'.

Parts and Relations

The first component of the representation is the *part*. A part may either be a *composite* part, composed of several subparts, or a *primitive* part that has no further decomposition.

The second component of the representation is the *relation*. A relation is a descriptive entity which specifies the structural relationship between two parts, and includes information such as relative size, position, and orientation. Two types of relation are distinguished, namely the *parent-subpart* relation between a part and one of its components, and the *neighbour-neighbour* relation between two parts that are connected, close, or otherwise structurally related in an 'interesting' way (such as alignment). The relationship may be described not only with respect to the frame of reference of each of the two parts, but also with respect to one or more common parent parts. For example, the distance between the two eyes of a humanoid in Figure 1 may be specified relative to the dimensions of the head as a whole.

Figure 2 shows an example of a very simple object and its description depicted as a graph. The boxes denote parts, heavy edges denote parent-subpart relations, and dotted edges denote neighbour-neighbour relations. The topmost part-node represents the object as a whole. It can be seen from this diagram that each part has three sets of relations: a set of neighbour-neighbour relations (called its nbrrels), a set of parent-subpart relations to its enclosing composite parts (called its parrels), and a set of parent-subpart relations to its subparts (called its subrels).

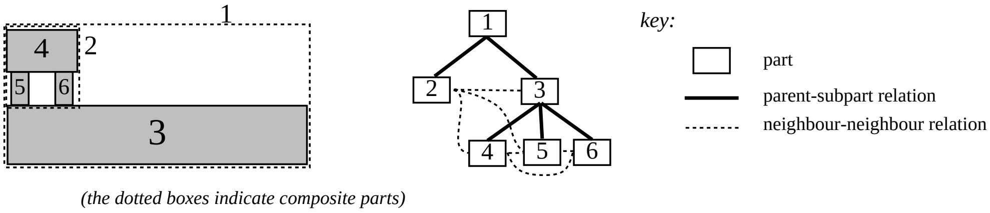

Figure 2: Parts and Relations

Parts and relations are characterised by attribute vectors, whose values may be of several different types, such as nominal, numerical, boolean, and so on. A generalised numerical value is represented as a range (min..max) or as a mean and variance. A generalised nominal value is represented as a set of symbols and their relative frequencies of occurrence within the contributing instance values. For example, out of 60 observed windows, 56 may be *rectangular*, 30 may be *square* (and rectangular), and 4 may be *round*.

Crels

The relations discussed above are between two parts within the same description. GRAM also allows relations between a part of a description and a part of some other concept description. In other words, a part can be related to "one of those", rather than to "that". For example, the description of the concept 'desk' may include relations between the legs of the desk and the concept 'floor'. This is interpreted to mean that "there exists a floor which is related to the legs of the desk in such and such a way". It does not specify a particular floor, as there is no explicit part within the desk description which represents the floor. Relations of this form are called 'concept relations', or crels. Similarly, a bedroom may be described in terms of crels to concepts such as 'bed', 'desk', and so on. However, because these components are not being explicitly represented in the description, relations between them cannot be made explicit. In the case of a 'bedroom' concept this may be acceptable since its typical components can be in many different arrangements.

The representation scheme also supports inheritence, where the details of the structure of a part may be inherited via an **ISA** link to a concept.

Structure and Context

Each part is described in terms of its *structure* and its *context*. Structure is specified by a set of properties (characterising its shape, aspect ratio, etc) and a set of subrels. Context is specified by another set of properties, a set of nbrrels, and a set of parrels. This explicit distinction between structure and context is useful because it enables a generalised description of a part to include a disjunctive set of structures or a disjunctive set of contexts. For example, the description of the back of a generalised 'chair' may include a disjunction of structures (each characterising a different variety of chair back). Similarly, the description of a generalised 'book' may include a disjunctive set of contexts, each characterising a different location that a book is most commonly found, such as on a bookshelf, on table, or in a person's lap. There are many such situations where context and structure are relatively independent, hence we consider it useful to be able to include this constrained form of disjunction in the representation.

Other work such as (Thompson and Langley, 1991; Connell and Brady, 1987; Brooks, 1981; Lebowitz, 1983; Winston, 1975) did not support this kind of disjunction, and therefore did not include the explicit distinction between structure and context in their representation.

Groups

Collections or sequences of similar parts can be represented as a single part whose structure includes a *group description.* A group description includes several properties characterising the group as a whole, such as its cardinality, the type of ordering of the group (whether a chain, loop, array, etc), and how similar the members are. It also includes a description of its *typical member* which is a generalisation of the individual members. Winston (1975) introduced typical members, in a simple form, and GRAM extends his idea in various ways: A typical member may be a complex composite structure, and may include relations between the parts of the typical member and other parts of the object, as in the case of a typical 'shelf' in a bookcase being related to the two side supports. The typical-member of a group is a concept in its own right. I call such a concept a *local-concept* because it is local to a particular object (in contrast with a *global-concept* which represents an independent generic object.) Therefore, other parts of the object may have crels to the parts of the typical member, such as shelf having a crel to the typical book on it. The typical member may also include generalised crels to its own components, and these represent the typical relationship between consecutive members of an ordered group (such as a row of books, or a stack of chairs).

Occurrence Counts

Each of the descriptive components of a concept (including parts, structures, contexts, relations, and groups) all have an *occurrence count* indicating how often it has been observed. This is used as a measure of the importance and optionality of each part and relation. For example, the occurrence count of the aerial of a generalised television set will indicate how many observed televisions had aerials on top of them.

3. The Matcher

To recognise an object as an instance of a concept, or to generalise a concept to include a new object, we must match the concept and the object. This involves finding the correspondences between the parts of the two descriptions. The process is a form of graph matching, where the nodes denote parts, and the edges denote relations.

The principal underlying the GRAM matcher is that the search is to be guided by the structure of the objects being matched, but not overly constrained by it. This achieves both efficiency and robustness.

Previous work such as (Dietterich and Michalski, 1981) did not make significant use of the description structure (*i.e.* the relations between parts) to guide the match, but instead took a more exhaustive approach. This is computationally expensive for large objects, since the possible combinations of part correspondences is exponential. For example, these systems could not easily cope with matching the two bicycles in Figures 3 and 4. The part-relation graph shown in Figure 5 has about 120 composite and primitive parts, so there are roughly 120! sets of correspondences between the parts of the two bicycles. (This is actually an underestimate since each pair of parts can be matched in several ways depending on which edges are put in correspondence).

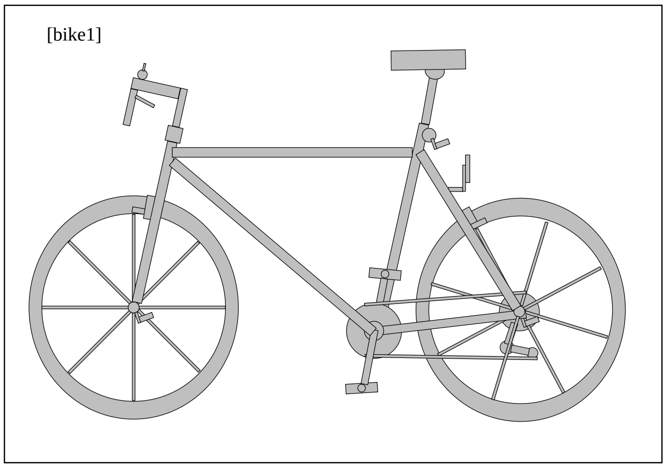

Figure 3: A Bicycle

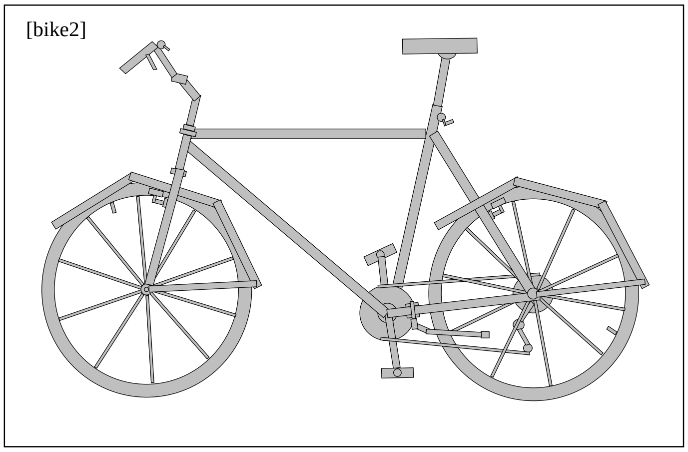

Figure 4: Another Bicycle

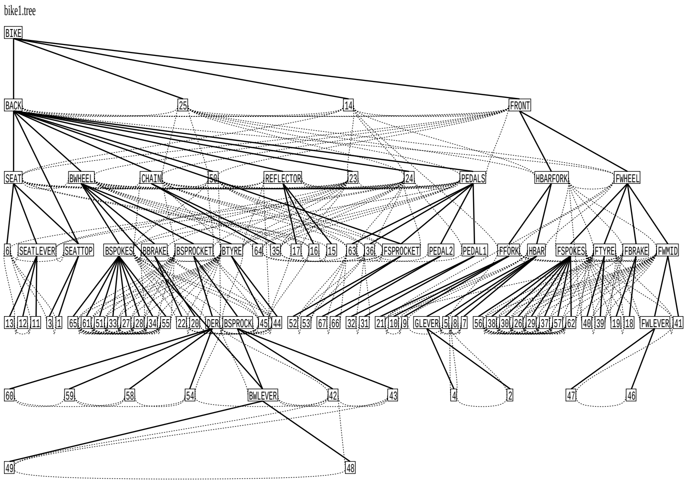

Figure 5: The part-relation graph for a bicycle

Other work, such as (Thompson and Langley, 1991; Connell and Brady, 1987; Brooks, 1981; Gennari et al., 1989), took the opposite extreme: the match process was tightly constrained by the part hierarchies, working from the root node of the two descriptions and finding correspondences by following strictly down the subpart links. If two similar objects were described with different decomposition hierarchies (such as having additional levels of partitioning) then these systems could not find the correct correspondences. Wasserman (1985) addressed this to some degree with his 'level hopping' technique. GRAM, however, takes a more flexible and robust approach.

Pnotes

The GRAM matcher describes each part-part correspondence in a *pnote*. A pnote specifies not only the two parts being corresponded, but also the edge correspondence, a score indicating structure and context similarity, and the status of the pnote. Three statuses are distinguished, namely *fixed*, *candidate*, and *conflicting*. A *fixed* pnote is one that has been selected as a correct correspondence; A *candidate* pnote is a proposed (or hypothesised) correspondence that is available for future selection; and a *conflicting* pnote is one that conflicts with a fixed pnote and is therefore no longer available for selection. The goal of the matcher is to find a complete and consistent set of fixed pnotes.

The similarity of the structure and context of two parts is computed by comparing their properties and relations (and perhaps the group descriptions of the structures). However, relations can only be compared if they involve parts that are already in fixed correspondence. Therefore, the score for a pnote is necessarily only a partial estimate until a complete set of fixed pnotes has been found by the matcher. Whenever a pnote is fixed, scores of other pnotes are updated if the parts are related to the parts of the newly fixed pnote. For example, if the correspondence between the pedals of two bicycles has been hypothesised, then fixing the correspondence between the frames will enable the relations between the pedals and the frame of each of the bicycles to be compared, thus enabling the score of the pedal pnote to be updated.

The "Propose and Select" Algorithm

GRAM's matcher algorithm, shown in Figure 6, is essentially an iterative cycle involving the proposal of new pnotes to be added to the set of candidate pnotes, and the selection of the best candidate to be added to the set of fixed pnotes. Before the cycle can begin, a 'seed' pnote is selected and fixed. GRAM currently uses the correspondence between the root parts of the two descriptions (*i.e.* the topmost part in each decomposition hierarchy). However, any other seed pairing can be chosen if there is strong confidence that it is a correct correspondence.

The first step of the cycle involves proposing new candidate pnotes on the basis of the most recently fixed pnote. This is done by three mechanisms called "spreading activation", "concept activation", and "distinct feature indexing". The best unambiguous candidate pnote is then selected, and added to the set of fixed pnotes. If no selection can be made, the conflict resolution process attempts to resolve ambiguities, unfix poor pnotes, or augment the descriptions with new partitions, in order to enable new correspondences to be selected. The search cycle continues until no more correspondences can be established. I will now describe some of the components of this algorithm in more detail.

"Spreading Activation"

In a top-down search (as in (Thompson and Langley, 1991; Connell and Brady, 1987; Brooks, 1981; Gennari et al., 1989)) a fixed correspondence of two parts leads to the proposal of cor-

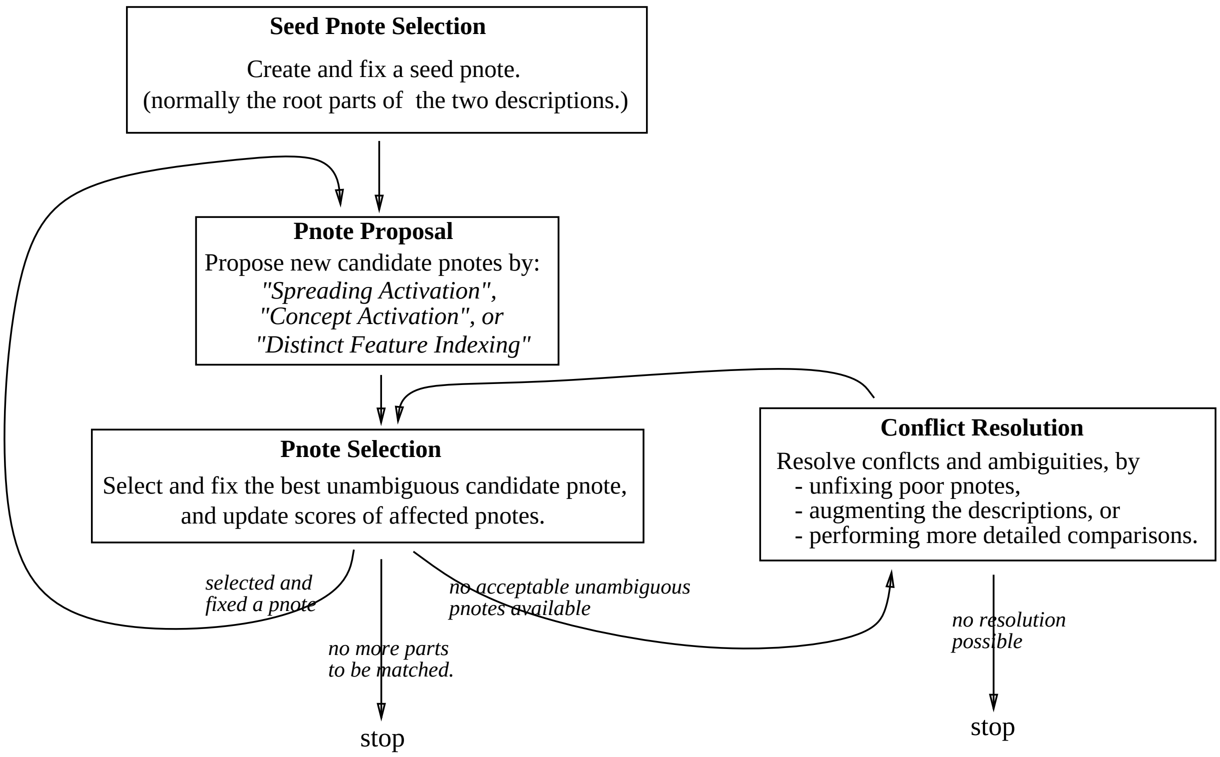

Figure 6: The "Propose and Select" Algorithm

respondences of their subparts. This approach fails if the two objects have been partitioned in different ways, since correspondences may be inaccessible via the decomposition hierarchies. The "spreading activation" mechanism resolves this limitation by not only proposing correspondences between subparts, but also between neighbours and parents.

This is illustrated in Figure 7. Suppose the pnote for parts A3 and B5 has just been fixed. The matcher then considers all pairings of the parents, all pairings of the subparts, and all pairings of the neighbours. If a pnote for a pairing does not already exist, and if the relations are sufficiently similar, then the part correspondence is evaluated. If the similarity is sufficiently high, then a candidate pnote is created.

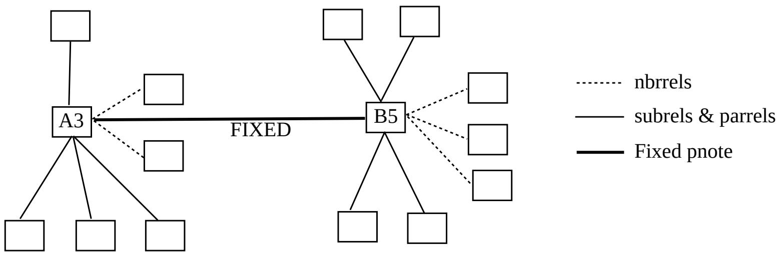

Figure 7: "Spreading Activation"

This mechanism enables the search to extend in any direction throughout the part-relation graph, rather than in a strict top-down fashion. Thus, even if two objects are partitioned in different ways, correspondences can usually still be found via neighbour-neighbour relations.

“Concept Activation”

The spreading activation mechanism only suggests correspondences of parts that are explicitly related to parts that have previously been fixed. In other words, the search is constrained by the graph structure and so will tend not to find correspondences of parts that share similar substructure but have different contexts, such as the two watches in Figure 1.

"Concept Activation" is a simple mechanism for proposing possible part correspondences without being constrained by the description structure. It is based on the observation “Oh, I've just seen one of those, maybe they correspond.” If we (as humans) look at Mary in Figure 1 we will probably notice an instance of the concept ‘watch’. When we then look at Jim we will also recognise a ‘watch’, and this will lead to the hypothesis that these watches correspond, and then perhaps to the generalisation that a humanoid may have a watch which must be located at either of two locations.

So, whenever a pnote is fixed, all descendents of one of the parts are scanned and classified (if they have not already been classified earlier). When a concept is recognised, a pointer to the instance is recorded in it, as a kind of ‘short term memory’. The subcomponents of the second part are then scanned and classified. If any concept that is recognised has pointers to a small number of components of the other part (*i.e.* if it has been ‘activated’) then correspondences may be proposed. For example, if the front wheels of two bicycles have been corresponded, then the concept activation mechanism may suggest a correspondence between the valves.

“Distinct Feature Indexing”

Another similar method for proposing correspondences between the subcomponents of two matched parts is based on a common feature value, such as colour, shape, aspect-ratio, structure profile, size, and so on. A common feature value can be found efficiently by creating a feature index for each of the two parts. If one (or a very small number) of descendents of each of the two parts share the same feature value, then these correspondences are evaluated in more detail to determine whether a candidate pnote should be created. For example, if one car has an aerial at the front right, and another car has an aerial at the back left, then (even when the concept ‘aerial’ has not been learned) the correspondence may be identified via feature indexing due to the distinctive long-narrow shape.

Feature Indexing is essentially equivalent to Concept Activation, except that instead of learned concepts being activated, feature values are activated.

Selecting the Best Candidate Pnote

At each iteration of the search cycle the best candidate pnote is selected and fixed. The way in which this choice is made is crucial to the success of the search. If a bad choice is made then it will have to be ‘unfixed’ later, and before this occurs it may cause a chain of further incorrect correspondences.

GRAM minimises the risk of error by computing a ‘choice-score’ for each pnote based on three factors: (a) The *similarity score* of the two parts, (b) the *completeness* of the similarity score, and (c) the *ambiguity* of the pnote.

The evaluation of the *similarity score* of two parts was discussed earlier. The *completeness* of a similarity score is an indication of how many of the parts' relations involve parts that are already in fixed correspondence and are therefore able to be compared. A high measure of completeness is favourable because it indicates that further fixing is unlikely to significantly lower the pnote score. The *ambiguity* of a pnote is a measure of how good it is relative to other pnotes with which it conflicts. The matcher should favour the selection of a pnote which has a significantly better similarity score than its competitors, since this minimises the likelihood that it will need to be unfixed later in the search. If the ambiguity of a pnote exceeds some threshold level, then it cannot be fixed even if it has strong similarity and completeness.

Conflict Resolution

When there are still parts to be matched but no pnotes can be selected, various techniques are applied to enable the matcher to continue.

Sometimes a pnote is incorrectly fixed, especially early in the search when there are only a few other fixed pnotes to provide supporting evidence. Such pnotes can be identified and unfixed when their similarity score drops below some threshold level, or when the scores of conflicting pnotes are no longer significantly lower. Hence 'unfixing' is one way of enabling the matcher to continue when it gets stuck. Cycles can be prevented by various detection mechanisms.

Another technique is to augment the descriptions with new parts in order to resolve ambiguities or to create correspondences for unmatched parts. Figure 8 illustrates a few of these situations. For example, the search finds ambiguous correspondences between the primitive part A4, and both the composite part [B5+B6+B7] and the primitive part B5. To resolve this ambiguity, a new composite part is created in the toy1 description, with A4 as its only subpart. A new pnote can then be created and fixed for the correspondences between this new part and the composite part in toy2, and between A4 and B5.

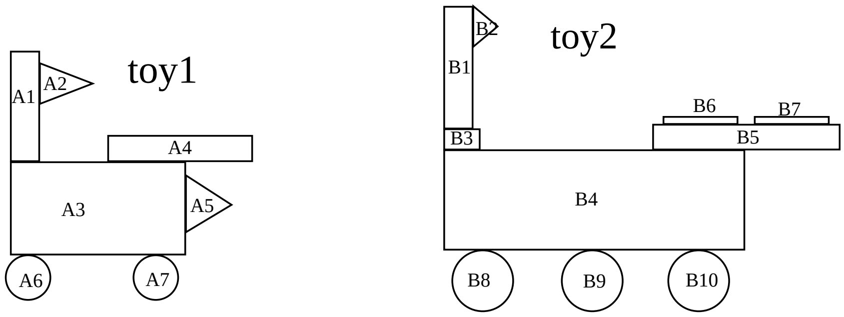

Figure 8: Toys

Ambiguities are also identified in the case of A6 and A7, and B8, B9, and B10. Assuming that the 'wheels' in toy2 are not already represented as a group, the ambiguity causes the matcher to create a new composite part in each of the toys, both being described as a group. A pnote can then be created to match these grouped parts. In other words, the group of two wheels in toy1 is put in correspondence with the group of three wheels in toy2.

A slightly different situation arises in the case of A1 and A2, and B1, B2, and B3. Supposing that A1 and A2 have not been represented as a single part, the matcher cannot find a correspondence for the composite part [B1+B2+B3]. However, because two of its subparts, B1 and B2, match well with A1 and A2, and because these two subparts form most of its solid area, the matcher creates a new composite part for A1+A2, hence enabling a correspondence to be created.

GRAM's generaliser (which is not described in this paper) will not necessarily incorporate such composite parts into the concept description, and may even remove previously existing compositions which are no longer useful. The matcher simply proposes the best correspondences it can find, and it is up to the generaliser to decide how to use them.

Another technique, which can be applied when all but a few parts have been successfully matched, is to test all pairings of the remaining unmatched parts. Their contexts will not match well (since otherwise pnotes would have been created via spreading activation) but their structure may match sufficiently well for a correspondence to be established. For example, if the "concept activation" technique could not find correspondences between the two watches in Figure 1 due to not knowing the 'watch' concept, the correspondence may still be found using this "match-the-leftovers" technique. The technique is not applied, however, to subparts of other unmatched parts, but only do the highest unmatched parts in the decomposition hierarchy. The unmatched watches actually consist of several unmatched parts, but only the correspondence between the whole watches is considered.

Ambiguities may sometimes arise because the scores of a pnote are insufficiently complete. If the context and overall properties of several parts in two descriptions are quite similar, then the best correspondence cannot be established until some of their substructure has been compared. In such a situation the matcher performs a recursive 'nested' match of the substructures of the ambiguous parts, in an attempt to resolve the ambiguity.

4. Summary and Conclusions

This paper has briefly outlined the representation scheme and the matching algorithm of GRAM, a system that matches and generalises two-dimensional representations of physical objects. The representation is based on *parts* and *relations*, organised in a decomposition hierarchy. Other features such as neighbour-neighbour relations, groups, the distinction between structure and context, occurrence counts, and disjunctions, give the representation a richness that supports effective matching and generalisation.

The matcher is a hill-climbing search based on a "propose and select" approach. At each step in the search cycle new part-part correspondences are proposed and added to a pool of candidates, and the best one is then selected. Conflicts and ambiguities are resolved when the search gets 'stuck'. This may involve augmenting the descriptions, or performing more detailed similarity evaluation. Backtracking may also be performed if necessary, but the way in which correspondences are selected minimises the risk of this occurring. Several mechanisms are used to propose candidate correspondences, thus achieving robustness, flexibility, and efficiency. "Spreading Activation" proposes new correspondences on the basis of a previously selected correspondence by considering pairings of the subparts, neighbours, and parents of the two parts. This enables correspondences to be found even when two similar objects have been partitioned into different decomposition hierarchies. "Concept Activation" proposes correspondences between parts that have both been recognised as belonging to the same concept. "Distinct Feature Indexing" involves creating an index from features to parts, and this index is used to identify possible correspondences.

GRAM's matcher has been partially implemented, such that it successfully finds correspondences between the bicycles in Figures 3 and 4. I am still completing the conflict resolution mechanisms, and have yet to test the matcher on a larger variety of situations.

References

Brooks, R. (1981). Symbolic reasoning among 3-d models and 2-d images. *Artificial Intelligence*, 17:285–349.

Connell, J. (1985). Learning shape descriptions: Generating and generalising models of visual objects. Master's thesis, Department of Electrical Engineering, MIT.

Connell, J. and Brady, M. (1987). Generating and generalising models of visual objects. *Artificial Intelligence*, 31:159–83.

Dietterich, T. and Michalski, R. (1981). Inductive learning of structural descriptions: Evaluation criteria and comparative review of selected methods. *Artificial Intelligence*, 16(3):257–94.

Gennari, J. H., Langley, P., and Fisher, D. (1989). Models of incremental concept formation. *Artificial Intelligence*, 40(1-3):11–61.

Lebowitz, M. (1983). Researcher: An overview. In *Procs of AAAI-83*, pages 232–5.

Marr, D. (1982). *Vision*. W.H. Freeman, San Francisco, CA.

Provan, G. M. (1988). Model-based object recognition: A truth maintenance approach. In *CAIA-88*, pages 230–5.

Thompson, K. and Langley, P. (1991). Concept formation in structured domains. In Fisher, D. H. and Pazzani, M., editors, *Concept Formation: Knowledge and experience in unsupervised learning*, chapter 5, pages 127–161. Morgan Kaufmann, San Mateo, CA.

Wasserman, K. (1985). *Unifying Representation and Generalisation: Understanding Hierarchically Structured Objects*. PhD thesis, Columbia University.

Winston, P. (1975). Learning structural descriptions from examples. In Winston, P., editor, *The Psychology of Computer Vision*, chapter 5, pages 157–209. McGraw-Hill, New York.