ISSN 0803–6489

Connectionist-based Analogical Mapping of Object-Oriented Specifications: A Representational Scheme

Barry Kristian Ellingsen*

REPORT NO. 59 MAY 1997

*Department of Information Science, University of Bergen, N-5020 Bergen, Norway. Email: barry.ellingsen@ifi.uib.no

Abstract

In recent years the focus on software component reuse has been elevated to an important research topic within software engineering circles. This is particularly visible among object-oriented communities. Although reuse is not new to object-oriented software developers, it has matured sufficiently to be applied in the early phases of the development life-cycle. In the ROSA (Reuse of Object-oriented Specifications through Analogy) project (Tessem et al. 1994) the aim is to apply analogical reasoning in the reuse of object-oriented specifications in the initial phase of the software development. This paper address distributed connectionist networks in the analogical mapping process of object-oriented specifications. OOram (Object-Oriented Role Analysis and Modeling) role models (Reenskaug et al. 1996) have selected for component specifications. The role models are represented as directed graphs in the mapping process. This paper shows that analogical mapping of OOram role models is not a trivial task. Experiments indicate that presenting the graphs as holistic patterns to a neural network in order to establish a mapping between the elements is difficult. In particular, it is difficult to find a uniform representation of the graphs that is sensitive to the structure. Experimentation with different distributed connectionist networks, combinations of structural and semantic information, and alternative coding schemes is necessary to obtain better results.

1 Introduction

As the software engineering field has matured over the years, it has been a growing interest in the reuse of software components. Today this is quite apparent in the object-oriented methodologies in which reuse is strongly motivated to enhance the productivity and quality of the software components. Further, reuse is no longer restricted to mere code since the object-oriented abstraction mechanisms facilitate reuse in the earlier phases of the software life-cycle (Korson and McGregor 1990; Fichman and Kemerer 1992).

Much research has been invested in design and implementation of reusable classes and class hierarchies (Johnson and Foote 1988), but less research has been invested in reuse of analysis and specification components. However, there is a growing belief in the object-oriented software engineering community that reuse ought to be the realm of analysis and design rather than code (Biggerstaff and Richter 1989). Moreover, since object-oriented design components contain no implementation details, they are more apt to reuse than mere code.

Successful component reuse is heavily constrained by the two dependent key factors *retrieval* and *representation*. First, reuse implies that relevant components must be retrieved from a collection of previous stored components. Second, there is a representational problem in that components must be described properly to be a subject for reuse. None of these problems are trivial and need to be addressed in software engineering environments. Several methods and techniques have been proposed to accommodate these problems, such as ITHACA (Fugini et al. 1992) and REBOOT (Morel and Faget 1993).

In the ROSA project (Reuse of Object-oriented Specifications through Analogy) the aim is to apply analogical reasoning in the reuse of object-oriented specifications (Tessem et al. 1994). The OOram (Object-Oriented Role Analysis and Modeling) role modeling technique (Reenskaug and Andersen 1992) was selected for specification of object-oriented components in the analysis phase. A role model is described along a set of dimensions, called *facets*, in order to support fast retrieval (Tessem 1995a). After the retrieval process of potential role models, called *base filtering*, a subsequent mapping process construct correspondences between elements of the retrieved role model (base analog) and the new role model (target analog). The later process is, in this context, crucial for transferring knowledge between the two domains.

This paper addresses the use of neural networks[1] in the analogical mapping between elements of a base analog and a target analog. The base analogs are role models retrieved in a base filtering process, and the target analogs are new role models. The main contribution in this work is to provide a *representational scheme* of the role models for a connectionist-based analogical mapping. Some problems related to analogical mapping of role models by distributed connectionist networks are also discussed. A two-phased process of analogical mapping is proposed, where the OOram role models are coded as *distributed representations*.

To begin with, in Section 2, the background for this work is presented; first a brief overview of two prevailing, but somewhat different, theories of analogical reasoning,

[1] In this paper I will use neural networks and distributed connectionist networks interchangeably, as opposed to localist connectionist networks (see Section 2.3).

followed by a description of OOram role models as candidates for analogical reasoning. Section 2 also presents an overview of different paradigms related to knowledge representation and processing. To what extent these paradigms can be unified is briefly discussed. This is followed by a presentation of the phases of analogical reasoning in the ROSA project. Section 3 provides a representational scheme of the role models in a graph notation. The purpose of the graph notation is to specify a framework for a connectionist-based analogical mapping of the role models in terms of labeled graphs. Subsequently, in Section 4, an outline of a connectionist-based analogical mapping that considers different constraints is presented. In Section 5 a two-phased algorithm is presented together with a description of the data set that is used. This is then contrasted with related work. Lastly, this paper closes in Section 7 by providing some concluding remarks and an outline for further research.

2 Background

2.1 Analogical reasoning

Broadly speaking, analogical reasoning (AR) is the ability to use a previous solution to an old problem as means for solving new and similar problems. The concept of analogy has been investigated by many researchers in several disciplines, and the contribution from psychology and philosophy is indispensable. However, the most promising work in our case is related to computational approaches to analogy. A substantial part of the theoretical investigations on computational analogy has its origin in the paradigm of traditional symbolic artificial intelligence. Therefore, several models of mental structures and processes have evolved, most of which assumes that there is a correspondence between the processes of human thoughts and the programs that run on computers (Thagard 1988; Haugeland 1986). This classical assumption has been challenged by the connectionist paradigm. For example, Holyoak and Thagard (1989) have studied the use of a connectionist network as a cooperative algorithm for analogical mapping. Each unit in the network represents a feature and the connections represent constraints among the features. This type of network is characterized as a *constraint satisfaction* network (Rumelhart and McClelland 1986a).

There is a broad consensus among researches that AR includes several phases. Already mentioned is the initial retrieval phase of candidates from a collection of analogs (base filtering), followed by a transfer phase. Analogical transfer can be further subdivided into a *mapping phase*, where correspondences between elements of the previous solution (base) and the new problem (target) is established, and an *evaluation/adaption* phase for generating a solution in the target domain. When experiences from previous analogies are considered to improve its performance on later analogies, then a subsequent *learning* process takes place.

Besides the work of Holyoak and Thagard, the structure mapping theory of Gentner (1983) has played an important role in the development of computational models of AR (Falkenhainer et al. 1989). These two theories have slightly different approaches to AR. The difference between the theories become apparent by investigating *representation* and *mapping*. The representation concerns how the knowledge is coded in a memory, whereas mapping concerns how correspondences

between the elements of the base and target are established. Both Holyoak and Thagard's constraint satisfaction theory and Gentner's structure mapping theory relies on symbolic representation of the knowledge in terms of predicate calculus. Moreover, both theories recognize mapping as the significant phase of AR. However, whereas Holyoak and Thagard use a connectionist network to constraint the mapping that includes structural, semantic, as well as pragmatic aspects (for example purpose and goal), Gentner's structure mapping theory focuses primarily on rules and structural constraints of the mapping process.

The retrieval and learning phases are not the major concern in this paper; we assume that the base analogs have been retrieved in a phase prior to the mapping. As neural networks are systems capable of learning, it is expedient to regard learning from two different angels. First, neural networks are capable of adapting to an environment, which makes learning an immanent and integral part of the system. On the other hand, learning in the sense of AR has a slightly different meaning in terms of abstracting general principles and concepts. Nevertheless, it is convenient to regard *mapping* and *learning* as separate phases of AR. This is not to say that a neural network cannot do both, but in this work I will concentrate on neural networks in the mapping process.

2.2 Role models and analogical reasoning

We have selected OOram role models for the ROSA project because they are more general than objects and therefore more apt to reuse than class and object design (Biggerstaff and Richter 1989). Another reason for using role models is that analogy between different specifications identifies more easily when the roles an object plays in an application are considered (Bjørnestad et al. 1994). Moreover, each role is characterized by the messages it can receive from and send to other roles, which, in turn, defines the model's collaborating structure (Reenskaug et al. 1996).

Gentner emphasizes that it is the relationships among the elements of the structures that convey the analogy, rather than simple surface features (Falkenhainer et al. 1989). Thus, the aim of the analogical mapping of role models in the ROSA project is to transfer knowledge from one model to another based on the model's collaborating structure.

As the role models grow in terms of the number of roles and messages, they become too large and complex for efficient reuse by analogy. Therefore, by restricting the reusable components to smaller submodels, we believe a greater potential for reuse is obtained. A submodel is a smaller part of the system showing the roles that take part in a particular activity. We believe that the formal and consistent notation of the OOram role models makes them well suited for reuse purposes.

2.3 Neural networks and analogical reasoning

There is an ongoing debate among AI researchers, cognitive scientists, and connectionists regarding the tenet of human cognition. The debate is often motivated by different philosophical assumptions. Aside from the hypothetical discussion regarding biological versus cognitive plausibility, and different philosophical doctrines, the most interesting debate from the perspective of this work is the design of *compu-*

tational models of human reasoning and cognition. Controversies are also apparent in the computational approaches to AR, where the debate often goes along two intimately related dimensions: *knowledge representation* and *knowledge processing*.

Two paradigms of knowledge representation

The first dimension, knowledge representation, splits the researchers in two opposite positions: those who are committed to a high-level symbolic representation of knowledge, and those who distribute the knowledge among a large number of highly interconnected processing units. The first position is held by the classical AI researchers and cognitive scientists, whereas the latter is advocated by the connectionists. However, most of the work of AR is largely motivated by the assumption that knowledge of the real-world (concepts, objects, events, plans, goals etc.) are complex entities that require a high-level symbolic representation. From an ontological point of view the symbolic paradigm assumes that the knowledge represented in a memory has a semantic interpretation similar to their real-world counterparts. These symbols have an atomic, discrete and static nature, and are often complex in terms of representing different types of declarative and procedural knowledge. Examples of such representations are frames (Minsky 1985), scripts, schemata (Schank 1980), and production rules (Newell and Simon 1972). As for the low-level sub-symbolic representational position, the knowledge units do not have similar semantic interpretations, nor have the weights that connect the units. Connectionist models do not make the strong distinction between different ontological categories. In contrast, the connectionists assume that knowledge emerge from the highly interconnected and rather simple units in the network.

The knowledge processing concerns how the knowledge is manipulated in a memory. Like the first dimension, knowledge processing divide the researchers in two positions: a symbolic-based and a connectionist-based. This distinction is further discussed below.

Two paradigms of knowledge processing

Over the years, two different views of knowledge processing have evolved: a *symbolic-based* and a *connectionist-based*. The first view represents the classical AI approach where symbols are combined into larger structures by means of combinatoric operations in a memory. In that respect it can be argued that symbolic models are in some sense *representation-oriented*. Connectionist-based models, on the other hand, are more *process-oriented* in that it seeks to explore the processes in humans to explain knowledge processing. At a low-level of representation connectionist-based models spreads activation among units in a network, where each unit represents knowledge in three different ways:

1. *localist connectionist network*; spreading activation among simple symbolic units to represent the semantic relationship between conceptual elements (semantic networks), and

2. *marker-passing network*; a special version of the localist connectionist network, where activation is spread among high-level symbolic units (frames, scripts etc.), and

3. *distributed connectionist network*; spreading activation among simple non-symbolic units (neurons).

The marker-passing network is probably closest to the classical symbolic AI paradigm, whereas the distributed connectionist network is closest to *neural networks*. Barnden (1994a) argues that connectionist-based AR is restricted to localist and marker-passing connectionism. According to Rumelhart and McClelland (1986b), a neural network does neither represent knowledge localized in particular units, nor do the units have any semantic interpretation; knowledge is, in contrast with localist and marker-passing networks, distributed among the highly interconnected units in the network. However, all three types of networks connect the units by means of scalar values (weights) in order to adapt to an environment.

Although Holyoak and Thagard (1989) use a connectionistic approach, their principal model of AR is a localist connectionist network. There are, however, models that embrace both the symbolic and the connectionistic dimension of knowledge representation and manipulation. For example, Eksridge (1994) and Kokinov (1994) combines the two dimensions into hybrid models of AR. In most these cases, a connectionist network is used to represent the associative relevance among the knowledge elements, that is, to maintain a long-term memory.

Symbolism and connectionism: A unified model of AR?

As discussed by several authors (e.g., Eksridge 1994; Kokinov 1994; Holyoak et al. 1994), connectionism contributes to a broader understanding of the diverse processes of AR, both seen from a cognitive and from a computational point of view. Moreover, as reported by Dinsmore (1992a), AR has the potential to close the gap between symbolism and connectionism. As tangible this thought might be, unifying connectionism into AR may have several obstacles.

A potential direction is to create unified models that combine symbolic and connectionist models. This direction, yet promising, may meet more problems than it intentionally try to solve. The reason for this is mainly the conflicting nature of the two paradigms. Let us take *learning* as an example: whereas distributed connectionist learning requires statistical generalization from many repetitions on a large number of examples, analogical learning systems often require rapid "two-trial" learning from a small number of instances. Moreover, combining high-level symbolic and low-level neuronal representation is not trivial simply because of the different levels of descriptions.

Although the symbolic paradigm has had success in AR, there are, nevertheless, several limitations are reported. As discussed by Dinsmore (1992b), connectionism seems to provide a better solution to many of the difficult problems that arise in symbolic models. Some of the problems are:

- dealing with imprecisely defined information
- approximate matching
- generalization
- soft constraints

- parallel processing

A major strength of connectionist networks is the ability to deal with imprecisely defined information. The performance of a connectionist system does not hinge on the accuracy and completeness of the information. Even in situations where missing or inaccurate information (noise) is presented, the connectionist network is still able to approximate a result. Moreover, given an incomplete pattern as input, the network is able to fill the gaps of missing information. This situation is often called *pattern completion*. As for the symbolic paradigm, if presented with noisy information, the model will most likely have a decreasing performance.

Presented with a sufficient number of training samples, a distributed connectionist network is able to do similarity-based generalization and categorization due to its rich internal representation. This is not easily obtained in the symbolic models, in which knowledge often is hand-coded and where the generalization abilities are limited.

Another important strength of distributed connectionist networks is the ability to implicitly identify and cope with multiple constraints. In a symbolic model the constraints are defined as rules, and as the number of rules and the complexity of the knowledge grows, the amount of computational overhead increases exponentially due to sequential search of the memory. A distributed connectionist network, on the other hand, does not have the same exponential scaling problem due to its continuous nature, and can cope with the constraints in parallel. The constraints appear as excitatory and inhibitory connections between the units in the network. The constraints are soft in that they can be overridden and modified continuously. Despite the conflicting nature of the two paradigms, taking advantage of the strength of connectionism, where symbolism is weak, is an appealing thought. An important question is therefore *how* to incorporate connectionism into AR? Whereas there are several examples of incorporating localist connectionist networks (e.g., Holyoak and Thagard 1989; Kokinov 1994; Eksridge 1994), there are fewer examples of incorporating distributed connectionist networks, i.e., neural networks (Lange 1992). There is a compelling reason for this, in that AR relies on a complex knowledge representation in order to represent complex real-world entities with their relational structure (Barnden 1994a). This is not easily obtained in neural networks.

2.4 Analogical reasoning and ROSA

The mapping is, as discussed in Section 2.1, only one of several phases of AR. Figure 1 illustrates how the phases of AR are organized in the ROSA project. First, a base filtering process retrieves potential base model candidates from the ROSA repository. The criteria for the base filtering are specified by the user in the target role model. Then, a connectionist-based mapping process generates a mapping model by establishing the appropriate correspondences between the elements of the target and base role models. Lastly, an adaption process generates a solution from the mapping model by transferring additional knowledge from the base role model to the target role model.

A possible learning process (not shown in Figure 1) can be applied to improve later retrieval and mapping processes. Although not investigated in depth here, we consider learning in ROSA to be the following:

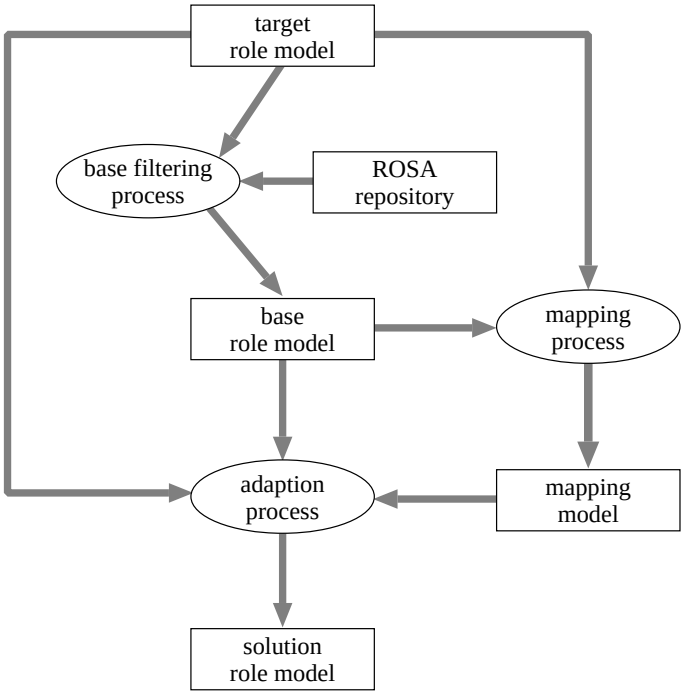

Figure 1: *A conceptual model of analogical reasoning in the ROSA project.*

1. Modification of the semantic relevance between the terms in the ROSA repository.

2. Storing generalized prototypes of the solution role models for abstracting common properties of the target and base role models.

The AR phases illustrated in Figure 1 are processed in a sequential order, with the base filtering process as the initial phase. The base filtering process and the mapping process is regarded as the most important computational processes in ROSA. A consequence of the sequential order of the phases is that the processes are to a certain extent independent of each other. Eksridge (1994) argues however that the phases of AR should be highly integrated. This is not the case in ROSA, where the phases are less integrated. Although this may not be a plausible model of human analogical reasoning, we consider the depicted model of AR as adequate for our purposes. The reason is that the solution relies on representation and processing in the different phases, i.e., symbolic representation and processing of the role models in the repository versus sub-symbolic in the neural network mapping process.

3 A representational scheme

This section presents a representational scheme of the OOram role models. The aim is to find an adequate representation of the role models that can work in conjunction with a neural network mapping process. In the following section it will be given an overview of the OOram role model notation, followed by a specification of the role models in terms of graphs and relation matrices.

3.1 OOram role model notation

An OOram role model is depicted in Figure 2, and illstrates a consultant enterprise that organizes their work in project groups. A role is represented as a super-ellipse, and each role is assigned a name. The collaborating structure of the roles is defined in terms of paths between the roles through which messages can be sent. A message path is denoted by a solid line between roles, and ports indicate the semantic relationship with the other role. A role can receive messages from and send messages to other collaborating roles.

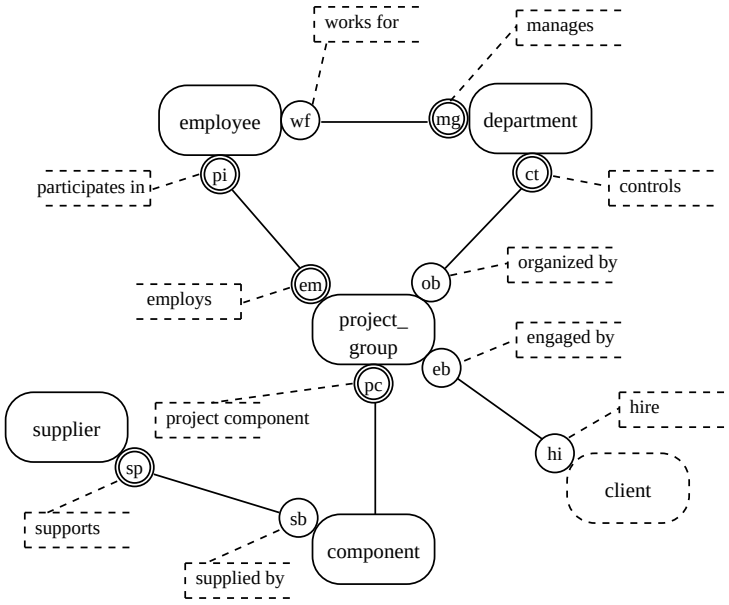

Figure 2: *OOram role model of a part of an enterprise.*

A port, shown as single or double circle connected to the role, defines the knowledge a role has about the remote role, that is, the cardinality of the relation. The notation of the ports is as following: a small circle denotes that the near role knows about exactly one collaborator; a double circle denotes that the near role knows about any number of the collaborator; a message path without a port denotes that the near role does not know the collaborator. For example, the port *ct* (controls) in the role model indicates that the role *department* knows about any number of the collaborating role *project_group*, whereas the port *ob* (organized by) indicates that the *project_group* role knows about only one collaborating *department* role. The *component* role however, does not know any of the collaborating *project_group* role, and therfore cannot send messages to the *project_group* role. All roles and ports have names that give them semantics. The message paths carry semantic information and denote the content of the collaboration. Another semantic information in the role model is the *stimulus role*, denoted by a dashed role. In Figure 2, *client* is the stimulus role that initiates an activity in the model.

3.2 Graph representation of role models

The role models can be given a graph representation as following. Let V be a finite non-empty set of vertices, each representing a role in the role model. Further, let $E \subseteq V \times V$, where $(v_1, v_2) \in E$ if and only if there is a path between the roles corresponding to v_1 and v_2 and there is a port near v_1 on this path. Then the pair (V, E) is a directed graph representation of the role model.

No isolated roles nor recursive message paths are allowed in a role model. Hence, the directed graphs are connected, but not necessarily strongly connected. The role model in Figure 2 is represented as a directed graph shown in Figure 3. In case of simplicity, the roles are labeled according to their first letter (except the *client* role, which is labeled l to avoid conflicts). Then, let $V = \{e, d, p, s, c, l\}$, and $E = \{$(e, d),(e, p),(d, e),(d, p),(p, e),(p, d),(p, l),(p, c),(l, p),(s, c),(c, s)$\}$.

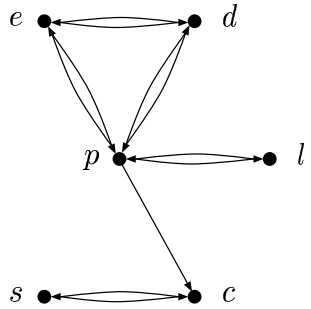

Figure 3: *The OOram role model in Figure 2 represented as the directed graph G.*

Generally, a role model carries more information than can be represented in a directed graph. For example, there is no distinction between "knows about one collaborator" (single circled port) and "knows about any number of collaborators" (double circled port) in the directed graph. The two role model components in Figure 4 show two different ports, where the upper state that role a knows about only one instance of its collaborating role b, and the lower state that role a knows about any number of instances of its collaborating role b. In the current graph notation both role models obtain an equivalent representation in the graph at the bottom of the figure. Moreover, the semantic property of the OOram's stimulus role (the vertex l in Figure 3) does not have a representation in the graph.

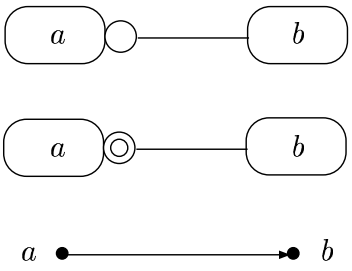

Figure 4: *Directed graph representation of role model ports.*

3.2.1 Role models as relation matrices

One way to provide the role models as input to the neural network is to transform the directed graphs to relation matrices. Given a directed graph $G = (V, E)$ representing a role model, where V is the set of vertices and E is the set of edges, the square $m \times m$ relation matrix $\mathbf{R}$ defines the relation matrix for E, where $|V| = m$. Each entry in the matrix $\mathbf{R}$ is defined by equation (1) below.

$$R_{ij} = \begin{cases} 1, & \text{if } (v_i, v_j) \in E \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

As the directed graphs contain no recursive relations, the matrix diagonal will always be zero. Moreover, the directed graphs are always connected, i.e., no isolated vertices, and the sum of the 1-values in row i and column i cannot be zero.

The directed graph shown in Figure 3 can be represented as the 6×6 relation matrix $\mathbf{R}$, shown in (2), where the vertices $V = \{e, d, p, s, c, l\}$ denote the rows and columns.

$$\mathbf{R} = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

4 Analogical mapping of role model graphs

Gentner (1983) argues that the structure of the analogs is crucial to the mapping process, where the elements of the analogs are placed in correspondence by virtue of their common relational structure. This is also dominant in our case, where the mapping is based on the correspondences between common relational structures of roles in the base and target role models. Thus, given two graphs, one target $G_t = (V_t, E_t)$ and one base $G_b = (V_b, E_b)$, we can define an analogy as a general mapping function M, as in Equation 3 below,

$$M : V_t \rightarrow V_b; \quad E_t \rightarrow E_b \tag{3}$$

where we, given the vertices and their relations in the target graph, find corresponding vertices and relations in the base graph. The vertex mapping function, $m(v_i) = v_j$, takes as argument the vertex v_i in the target graph and return the corresponding vertex v_j in the base graph. Likewise, the edge mapping function, $m((v_i, v_j)) = (v_k, v_l)$, takes as argument a target edge (v_i, v_j) and returns the corresponding base edge (v_k, v_l).

Given the mapping function M, we want, ideally, to obtain a *one-to-one* correspondence between the vertices in the two graphs over an equal set of relations. A major problem in analogical mapping is to find the corresponding vertices in the

two graphs *without* considering the entire set of possible correspondences. A remedy to this problem is to impose different types of constraints on the mapping. The type of analogical problem solving we are contemplating in the ROSA project would be too restrictive if we impose the restriction of isomorphism on the graphs to be mapped. For example, a typical reuse situation is to provide an incomplete specified role model as target, and then augment the target with a previous retrieved base role model to obtain a solution to the problem (see Figure 1 for details).

Apparently it is not strictly necessary to have an isomorphism between the two analogs, in that the base and target analogs can have an unequal number of roles and relations; some roles and relations in the two analogs need not have any correspondences at all. Therefore, as pointed out by Owen (1990), a *partial homomorphism* is a more realistic constraint, that is, mapping of as large subsets of the two graphs as possible. Other types of constraints are discussed later.

4.1 Structural information

The AR phases in the ROSA project, shown in Figure 1, indicate that the mapping process has two inputs: the specified target role model and the retrieved base role model from the repository. A mapping model is provided as output of the mapping process. Figure 5 depicts a neural network-based mapping process, taking the base and target relation matrices as input and generates a correspondence matrix as output. The correspondence matrix is the mapping model shown in Figure 1.

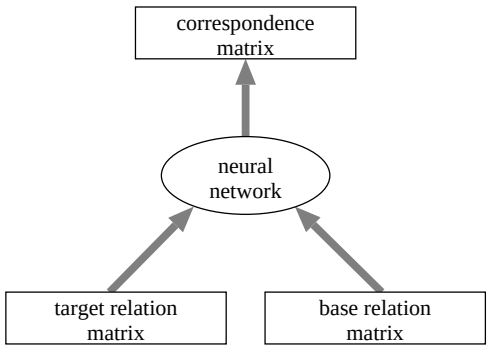

Figure 5: *Neural network-based mapping process.*

Essentially, we want the neural network to learn the mapping between the elements of two graphs, one base and target. For example, given a directed graph of a target role model $G_t = (V_t, E_t)$ and a base role model $G_b = (V_b, E_b)$ shown in Figure 6, we want a neural network to learn the mapping from the target to the base.

There are several similar substructures found in the two graphs in Figure 6. The symmetric relationship between vertex t_3 and t_5 in G_t corresponds to the symmetric relationship between vertex b_5 and b_2 in G_b. Thus, the vertex mappings $m(t_3) = b_2$ and $m(t_5) = b_5$, and the corresponding relation mappings $m((t_3, t_5)) = (b_2, b_5)$ and $m((t_5, t_3)) = (b_5, b_2)$ are established. In addition to the structural constraints, different semantic constraints can be imposed on the mapping process. We impose the restriction that a role model must have one, and only one, stimulus role. An

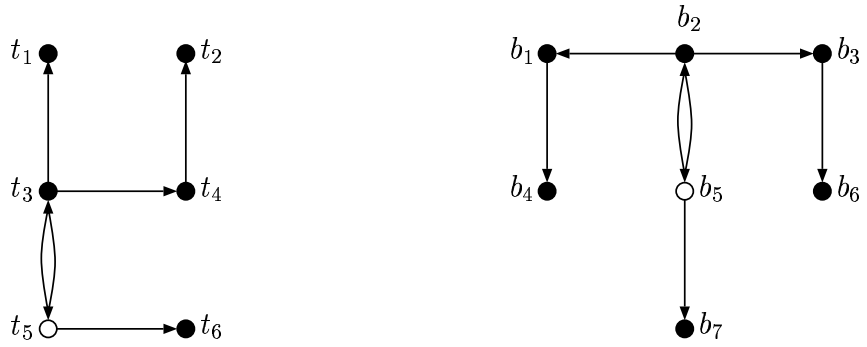

Figure 6: *Graph representation of base and target role model, where the left is the target graph G_t and the right is the base graph G_b.*

example of a semantic constraint is the annotation of the stimuli roles in the two graphs. Given that vertex t_5 represents the stimulus role in G_t, and vertex b_5 represents the stimulus role in G_b (denoted by unfilled vertices in the graphs), there is a high probability to establish a mapping between the two vertices, i.e., $m(t_5) = b_5$.

4.2 Network output: The correspondence matrix

Given the two graphs G_t and G_b in Figure 6, where $|V_t| = m$ and $|V_b| = n$ denotes the number of rows and columns respectively, the $m \times n$ correspondence matrix $\mathbf{C}$ is defined as in Equation 4 below:

$$C_{ij} = \begin{cases} 1, & \text{if } m(t_i) = b_j \\ 0, & \text{otherwise} \end{cases} \tag{4}$$

The relation matrices $\mathbf{R}_t$ and $\mathbf{R}_b$, shown in (5) below, for the two graphs in Figure 6 represent patterns for input to a neural network, and the associated $m \times n$ correspondence matrix $\mathbf{C}$ represents the output pattern. The set of mappings from target to base is defined as $\mathcal{R} = \{(t_1, b_1), (t_2, b_6), (t_3, b_2), (t_4, b_3), (t_5, b_5), (t_6, b_7)\}$, from which the associated correspondence matrix $\mathbf{C}$ is defined. Note that vertex b_4 in graph G_b has no corresponding mapping in graph G_t. Therefore, column 4 in the correspondence matrix $\mathbf{C}$ in (5) is zero.

$$\mathbf{R}_t = \begin{bmatrix} 0&0&0&0&0&0 \\ 0&0&0&0&0&0 \\ 1&0&0&1&1&0 \\ 0&1&0&0&0&0 \\ 0&0&1&0&0&1 \\ 0&0&0&0&0&0 \end{bmatrix}, \mathbf{R}_b = \begin{bmatrix} 0&0&0&0&1&0&0 \\ 1&0&0&0&0&1&1 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&0&0&0&0&0&0 \\ 0&1&1&0&0&0&0 \\ 0&0&0&1&0&0&0 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} 1&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0 \\ 0&1&0&0&0&0&0 \\ 0&0&1&0&0&0&0 \\ 0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&1 \end{bmatrix} \tag{5}$$

There are certain constraints that can be imposed on the mapping. A *many-to-one* mapping is regarded as valid, that is, several target vertices are mapped to the same base vertex. A *one-to-many* mapping, that is, the same target vertex is mapped

to several base vertices, is *not* regarded as valid as it violates the formal definition of a function. However, this restriction may occasionally be violated as we cannot always obtain the ideal one-to-one mapping on pair of isomorphic graphs. Also, a vertex in the target graphdoes not necessarily have to be mapped to a base graph vertex. These constraints are visible in the correspondence matrix $\mathbf{C}$ by requiring no more than one 1-value per row, whereas any number of 1-values are allowed in the columns. A zero-valued column indicates that the base vertex does not have a mapping in the target at all, as the case for the base vertex b_4 in graph G_b in Figure 6, where the fourth column in $\mathbf{C}$ is all zero-valued. The correspondence matrix $\mathbf{C}$ in (7) is an example of a *many-to-one* mapping where the same target vertex is mapped to two base vertices.

Feeding a neural network with the relation matrices shown in (5) as input pattern yields an input vector of $|V_t|^2+|V_b|^2 = 85$ input elements, and $|V_t| \times |V_b| = 42$ output elements. In a real-world system of ROSA, one must assume a certain variance with respect to the number of roles in the role models, hence the number of vertices in the graphs. Firstly, the role model variances may have implications for the analogical mapping process in that a one-to-one mapping cannot be obtained. Secondly, variances may also cause problems with respect to the neural network performance in terms of variable-sized input patterns. This is further discussed below.

4.2.1 Mapping variances

If we consider the variances with respect to the mapping, three different cases are identified:

1. Each vertex in the target graph is mapped to a corresponding vertex in the base graph. This is the ideal *one-to-one* mapping, but not necessary isomorphic, situation where $|V_t| = |V_b|$. An example is the graphs shown in Figure 7, where $G_t = (V_t, E_t)$ is the target graph with vertices $V_t = \{t_1, t_2, t_3\}$ and the set of edges $E = \{(t_3, t_1), (t_3, t_2)\}$, and $G_b = (V_b, E_b)$ is the base graph with vertices $V_b = \{b_1, b_2, b_3\}$ and the set of edges $E_b = \{(b_3, b_1), (b_3, b_2), (b_2, b_3)\}$. The stimulus role vertex t_3 in the target graph is mapped to its counterpart vertex b_3 in the base graph, i.e., $m(t_3) = b_3$. The correspondence matrix for this mapping is shown in (6) below.

$$\mathbf{C} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

   Note that the two graphs are not isomorphic in that there is no equivalent to the edge (b_3, b_3) in the target graph, that is, $(t_2, t_3) \notin E_t$.

2. Two or more vertices in the target graph are mapped to the same vertex in the base graph. An example of this is illustrated in Figure 8, where $G_t = (V_t, E_t)$ is the target graph with vertices $V_t = \{t_1, t_2, t_3, t_4\}$ and the set of edges $E_t = \{(t_1, t_3), (t_3, t_4), (t_4, t_1), (t_4, t_2\}$, and $G_b = (V_b, E_b)$ is the base graph with vertices $V_b = \{b_1, b_2, b_3\}$ and the set of edges $E_b = \{(b_1, b_2), (b_1, b_3), (b_3, b_1)\}$. The two vertices t_1 (stimulus role) and t_3 in the target graph are mapped to

one and the same vertex b_3 (stimulus role) in the base graph. Further, the two mappings $m(t_4) = b_1$ and $m(t_2) = b_2$ are also correctly established, which provide the correspondence matrix shown in (7) below as the final mapping.

$$\mathbf{C} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix} \tag{7}$$

3. It is not a requirement that all base vaertices have to be mapped. This is the case for vertex b_4 in Figure 6, which is not mapped at all. The correspondence matrix is shown in (5) above.

Figure 7: *One-to-one mapping of the graph vertices, where the left is the target graph G_t and the right is the base graph G_b.*

Figure 8: *Many-to-one mapping of the graph vertices, where the left is the target graph G_t and the right is the base graph G_b.*

4.2.2 Pattern variances

Typically, neural networks do not handle symbolic structures, such as the directed graphs, particularly good. If there are variances in the graphs, as discussed in the previous section, the problem for the neural network may increase further. The problems are related to the network's relatively static structure[2]. Consequently, in a fixed structure, if arbitrary length patterns are presented to the network there might be superfluous units in the network that do not contribute in the learning process. Assignment of some kind "don't care" values are often required in such cases. A common remedy to this problem is to transform the patterns to a uniform representation. This solution works best if the patterns are insensitive with respect to the element's relational structure. For example, a gray-scale image can be transformed

[2]Although there are some neural network models that dynamically add and remove units during learning, the number of input and output units is, nevertheless, commonly constant.

with respect to translation, rotation, scaling and size without a significant loss of the information contents. However, this is not a satisfactory solution with graphs where the patterns are highly sensitive to the element's relational structure. To illustrate the difference between unstructured and structured compositional patterns, let us use a binary image matrix and a graph's relation matrix as an example: Inverting one entry (pixel) from *one* to *zero* in the binary image matrix does not change the global features of the image significantly. One the other hand, inverting one entry in the graph's relation matrix removes or adds an edge between two vertices in the graph. The later case may change the graph structure significantly. For example, removing an edge may cause the isolation of a vertex which, in turn, violates the connectedness of the OOram role model graphs.

Provided that the number of input/output units in the neural network is fixed, a possible solution to the variance problem is to constrain the number of vertices in the graphs. A maximum number of vertices can be defined as a constant k, from which a fixed number of input/output units in the network can be defined. For a graph $G = (V, E)$, if $|V| < k$, then temporary vertices can be added to the graph in order to obtain a fixed-size representation. A simple heuristic is to select a terminal vertex and add a l-chain (Tessem 1995a) to expand the graph to the maximum number of vertices k. l is the number of vertices in the chain and is defined as $l = k - |V|$.

To illustrate this solution, let us use the two graphs in Figure 6, one target graph $G_t = (V_t, E_t)$ and one base graph $G_b = (V_b, E_b)$, as an example. A neural network takes as input two graphs, one target and one base, with maximum $k = 10$ vertices. This yields a 10×10 correspondence matrix as output. In order to complete the graphs in Figure 6, a 3-chain with the vertices x_1, x_2, and x_3, is added to G_b, and a 4-chain,with the vertices y_1, y_2, y_3, and y_4, is added to G_t. This is depicted in Figure 9. Notice that the l-chains are added to an arbitrary terminal vertex in the graphs, i.e., vertex b_6 in G_b and vertex t_2 in G_t.

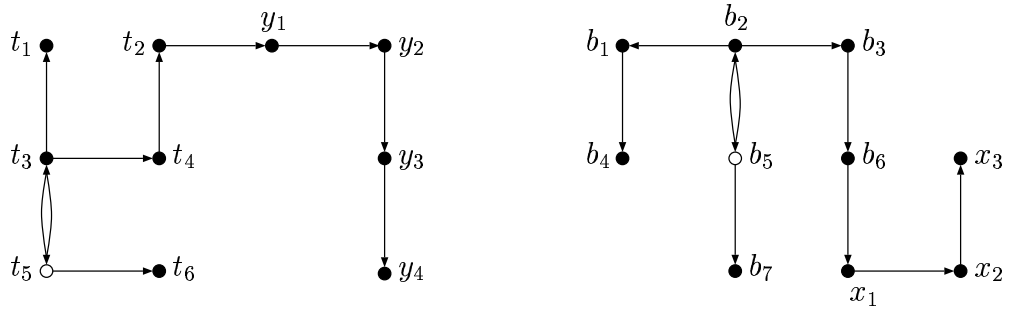

Figure 9: *Adding l-chains to complete the graphs. The left is the target G_t and the right is the base G_b.*

According to this simple heuristic we obtain a fixed size of the input relation matrices as well as the output correspondence matrix. Although we obtain a fixed size of the input graphs, it does not necessarily mean that a *one-to-one* mapping is obtained. For example, from the graphs in Figure 9 we may establish correctly the mapping $m(y_1) = x_1$, $m(y_2) = x_2$, and $m(y_3) = x_3$, whereas there is still no corresponding mapping for vertex b_4 in graph G_b. Matric $\mathbf{C}$ in (8) shows the mapping from the target to the base graph, where the rows denote the target graph vertices

$V_t = \{t_1, t_2, t_3, t_4, t_5, t_6, y_1, y_2, y_3, y_4\}$ and the columns denote the base graph vertices $V_b = \{b_1, b_2, b_3, b_4, b_5, b_6, b_7, x_1, x_2, x_3\}$. The zero-valued 4'th column indicates the non-mapped vertex of vertex b_4. Similarly, the 10'th row indicates that vertex y_4 in graph G_t has no mapping in G_b.

$$\mathbf{C} = \begin{bmatrix} 1&0&0&0&0&0&0&0&0&0 \\ 0&0&0&0&0&1&0&0&0&0 \\ 0&1&0&0&0&0&0&0&0&0 \\ 0&0&1&0&0&0&0&0&0&0 \\ 0&0&0&0&1&0&0&0&0&0 \\ 0&0&0&0&0&0&1&0&0&0 \\ 0&0&0&0&0&0&0&1&0&0 \\ 0&0&0&0&0&0&0&0&1&0 \\ 0&0&0&0&0&0&0&0&0&1 \\ 0&0&0&0&0&0&0&0&0&0 \end{bmatrix} \tag{8}$$

4.3 Semantic information

Although the mapping of OOram role models is constrained by structure, semantic information can be included to constrain the mapping furher. In this section it will be discussed how semantic information can be utilized for additional constraints on the mapping process. Generally, semantic information impose the constraint that the elements in the two analogs must have similar meaning, that is, share common properties in order to be mapped. The motivation for imposing semantic constraints is therefore to obtain a more effective mapping. In the matter of the ROSA project, a semantic constraint may indicate that the mapped roles and relations must have similar meaning or share some common properties in order to be mapped.

The stimulus role, discussed in Section 4.1, is another example of a semantic constraint. The stimulus role provides an additional meaning for a role by initiating the activity in a role model. Thus, there is a relatively high probability of establishing a mapping between the stimuli roles. However, as discussed in Section 3.2, neither the stimulus nor other types semantic information have any representation in the current scheme. The current scheme can be expanded to include semantic information as illustrated in Figure 10.

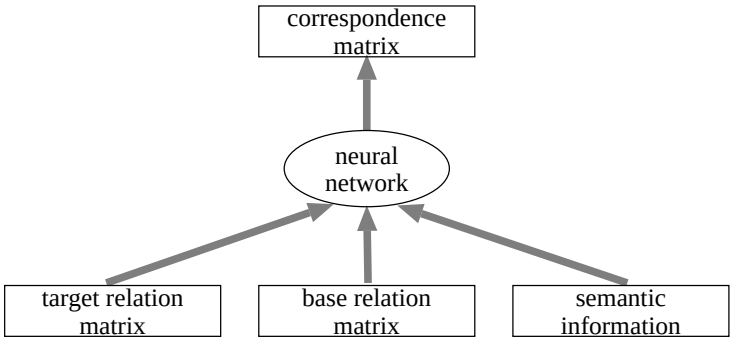

Figure 10: *A conceptual view of a neural network-based mapping process that includes semantic information.*

Some remarks are needed to explain how the semantic matrix in Figure 10 can be exploited to improve the mapping process. Each component category, listed in Table 1, is described in terms of one or more *facets* (Tessem 1995b). This yields a multi-dimensional description of the components which can be used in the retrieval of potential candidates.

component category	*facets*
full role model	name stimulus role
scenario	structure-description
submodel	name stimulus role structure-description
role	name port
message	name sender receiver

Table 1: *Component categories and their facets.*

In the ROSA project it is has been identified five component categories. However, as discussed previously, full role models are too complex for efficient reuse. As discussed in Section 2.2, the most promising component category for analogical retrieval and mapping is submodels, and perhaps *scenario*. It will be focused on submodels. Each submodel includes role and message components. The submodel's stimulus role is already emphasized as an important semantic constraint. Roles are important in that the name and port facets are throughly used in the retrieval process and in the subsequent mapping process. For example, the name indicates to what extent the roles share common properties in the base and target domain. Similarly, the port indicates the role's collaboration with other roles. Although a role's different ports cannot be fully represented in a graph (exemplified in Figure 4), their diversified meaning can nevertheless be embedded as semantic information.

Each facet stored in the ROSA repository has a related term space organized similar to semantic networks. Components whose terms in the target and base analogs are semantically related, are likely to be mapped to each other. A semantic relation is a matter of closeness between the terms. Such closeness is specified by a weight between the terms in the term space. Close related terms obtain a high value on the weight, whereas unrelated terms obtain a low value on the weight.

Returning to the conceptual mapping process shown in Figure 10. Semantic information can be represented in a matrix that reflects the similarities between the elements in the target and base model. In fact, several facets can be combined to obtain a comprehensive representation of the semantic information. If $|V_t| = n$ and $|V_b| = m$, then $\mathbf{S}$ is a $n \times m$ matrix whose elements are $0 \leq S_{ij} \leq 1$ and indicate the semantic similarities between the elements in the role models. A high value, close to 1, indicates a high semantic similarity and strengthens the belief that the elements should be mapped. Conversely, a low value, close to 0, indicates a low semantic

similarity and weakens the belief that the elements should be mapped. A small example illustrates the rationale behind this, where a target graph G_t is mapped to a base graph G_b, shown in Figure 11.

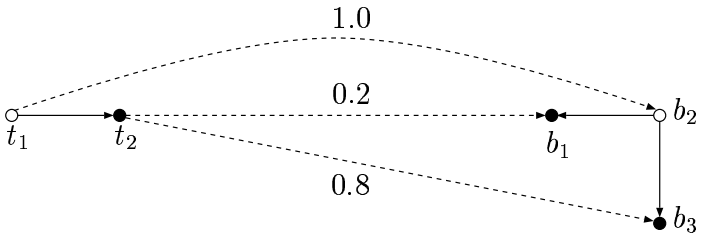

Figure 11: *Mapping between two graphs, one target G_t (left) and one base G_b (right), supported by semantic information.*

Provided that vertex b_2 in the base graph and vertex t_1 in the target are identified as stimulus roles, a high value, e.g., 1.0, is assigned to the particular entry in the semantic matrix. Therefore, the mapping $m(t_1) = b_2$ is established. Potentially, vertex t_2 in the target graph can be mapped to both vertex b_1 and vertex b_3 in the base graph. The values in the semantic matrix $\mathbf{S}$ suggest that semantic closeness between vertex (t_2, b_3) is higher than between (t_2, b_1). Therefore, it is more likely that the mapping $m(t_2) = b_3$ precedes $m(t_2) = b_1$. The semantic information for the mapping between the graphs in Figure 11 is given in matrix (9) below, where the rows denote the target graph vertices $V_t = \{t_1, t_2\}$ and the columns denote the base graph vertices $V_b = \{b_1, b_2, b_3\}$.

$$\mathbf{S} = \begin{bmatrix} 0.0 & 1.0 & 0.0 \\ 0.2 & 0.0 & 0.8 \end{bmatrix} \tag{9}$$

Potentially the semantic information obtained from the repository can be included as constraints in the neural network mapping process. In the example above, only the role name facet and the stimulus role facets are provided as semantic information. If several facets from different component categories are combined, a somewhat more complex situation arises. One possible solution is to use the theory of evidence, where the semantic similarities between the component categories are defined as belief functions, and combined by Dempster's rule (Shafer 1976) to obtain a comprehensive semantic matrix.

4.4 Combining structural and semantic information

One way to combine semantic and structural information is to use the structural information as constraints for modifying the semantic information. This strategy is schematically shown in Figure 12.

Because the semantic information a tentative mapping from the target to the base role model based on their semantic similarity, the strategy is to learn the neural network to modify the semantic matrix by using the structural information as constraints to obtain a mapping scheme according to the corresponding matrix

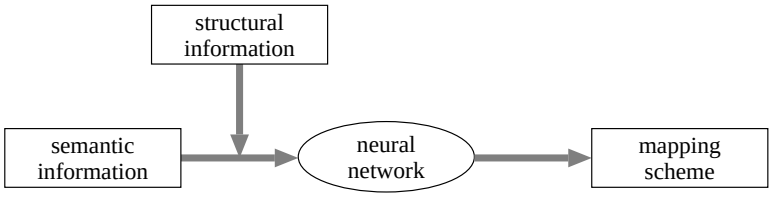

Figure 12: *Combining semantic and structural informaion.*

specified in Section 4.2. Hence, there will be, as shown in Figure 10, three input matrices: (1) the target relation matrix $\mathbf{R}_t$, (2) the base relation matrix $\mathbf{R}_b$, and (3) the semantic matrix $\mathbf{S}$. Likewise, there will be one output matrix $\mathbf{C}$. The correspondence matrix can be regarded as a modified semantic matrix, where the entries have been transformed to the binary values 0 and 1.

4.5 Subprocesses of analogical mapping

Analogical mapping may be separated into several phases, each carrying out a particular subtask. This is illustrated in Figure 13. First, the role models retrieved in the base filtering process along with a target role model need to be preprocessed according to the scheme described in Section 3.2 before presented to the neural network. Further, semantic information extracted in the base-filtering process is formatted and organized in matrices as described in Section 4.3. The preprocessing encode the graph structures and organizes the semantic information in numerical vectors for input to the neural network. After the neural network has processed the patterns, the output correspondence matrix need to be post-processed in order to obtain a complete mapping scheme. Post-processing includes preparation of the output from the neural network according to the constraints discussed in Section 4.1, including transfer of additional elements from the base to the target.

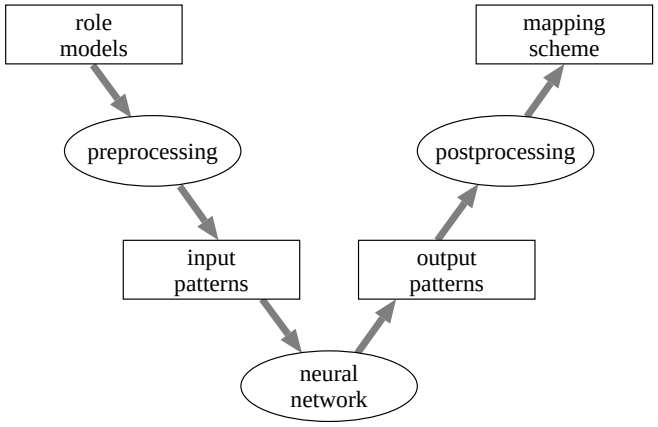

Figure 13: *Subprocesses of analogical mapping.*

The proposed model of analogical mapping must be regarded as hybrid, because there are both symbolic and connectionist aspects to it. Whereas the role models

presented to the mapping model have symbolic representations in terms of directed graphs, neural networks rely on distributed connectionist representation. A challenge is to investigate whether these to "paradigms" can be unified in a coherent model.

5 Experiments

The goal of the ROSA project is to design a tool for reusing object-oriented specifications. Techniques from AI have been found promising in that respect, where analogical reasoning is applied in the reuse phase. OOram role model components constitute the specifications of the problem domain, stored in a repository. In a base filtering process, potential base role models are retrieved from the repository for augmentation with a new role model (target) in a subsequent neural network mapping process. The analogical mapping we are contemplating here is a complex task, and is approached empirically in order to find potential solutions. The data set on which the neural network mapping model is tested, is presented in Section 5.1 below. Note that the role models used in the experiments are subject to the representational restrictions defined in Section 3.

5.1 Data set

In a real-world system of ROSA the number of role models stored in the repository can vary from a few to several hundred. Currently it is not a sufficient number of role models in the ROSA repository for the experiments. Therefore, data for the experiments are synthetically generated (Tessem 1996). The graph generation algorithm generates rooted, directed graphs between 3 and 20 vertices. The random generator simulates message passing either to new vertices or to already made vertices, and the result is graphs that are satisfactory similar to realistic models. It is also possible to generate random analogues to a graph by adding or deleting some edges to the graph. This *wriggling* algorithm gives us graphs that are very much alike the original graph. For every generated pair of graphs, one base and one wriggled target graph, mapping function M is provided. The mapping function M defines an "ideal" mapping from a structural point of view.

Given a mapping M, we can define a correspondence matrix $\mathbf{C}$, whose rows denote the target graph vertices and the columns denote the base graph vertices. The correspondence matrix $\mathbf{C}$ is defined in Equation 4.

A semantic matrix $\mathbf{S}$ is randomly generated from $\mathbf{C}$. The elements of $\mathbf{S}$ represent similarities between vertices in the two graphs. Similarity between vertices in the two graphs is initially assumed to be perfect (i,e., 1.0) between vertices that we consider match, and 0.0 for non-matching vertices. Then, to obtain the semantic matrix $\mathbf{S}$, noise is distributed on $\mathbf{C}$ using a random triangular distribution from 0 to 1 with top at 0. For non-matching vertices the noise is added, and for matching vertices it is subtracted. For the root vertices (i.e., the stimuli roles), the noise is scaled by a factor of 0.25. The following parameters can be used to control the graph generation:

- number of base graphs

- number of analogous target graphs for each base graph
- minimum and maximum number of vertices in the graphs
- number of changes in the analogue graphs

An annotation of the root vertex (stimulus role) is provided for all the generated graphs. It is assumed that the graph generation algorithm provides a sufficient basis for experimentation with different models and solutions.

5.2 Results

As the patterns are treated holistically, the network may have problems to discriminate among the elements of the graphs. Thus, correspondences between the graphs are not easily found due to lack of explicit representations of the graph elements. Nevertheless, the network can learn to find a mapping between pair of graphs, one target and one base, as holistic patterns, but the level of generalization may suffer significantly. This means that the network's ability to find a mapping for novel graphs is limited. Moreover, as discussed is Section 4.2, finding a uniform representation of the graphs in terms of relation matrices is difficult simply because small variances in the patterns may imply significant change in the graph structure.

In order to test a neural network on the analogical mapping task as described in this paper, 38 unique graphs with 3 vertices were generated. The first graph generated is the base graph, G_b, and the successive 37 graphs, $G_{t_1}, G_{t_2} \ldots, G_{t_{37}}$, are analogue target graphs with maximum two changes relatively to the base graph. Each vertex in the graphs is assigned unique labels from the set $\{t_1, t_2, t_3\}$. The graphs are represented as the 3×3 relation matrices $\mathbf{R}_b, \mathbf{R}_{t_1}, \mathbf{R}_{t_2}, \ldots, \mathbf{R}_{t_{37}}$. In addition, 3×3 correspondence matrices $\mathbf{C}_{(b,t_1)}, \mathbf{C}_{(b,t_2)}, \ldots, \mathbf{C}_{(b,t_{37})}$ are generated that define the mapping between the vertices of the target graph and the base graph. Examples of the generated graphs are shown in Figure 14.

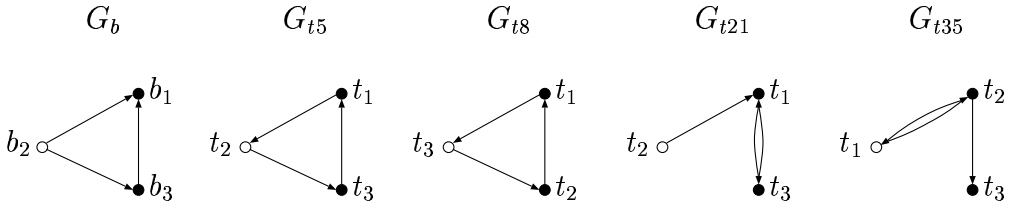

Figure 14: *Sample graphs from the input data.*

The unfilled vertices in the graphs denote the stimuli roles. In the current experiment framework it is not obvious how to embed the semantic information as discussed in Section 4.3. Therefore, for the case of simplicity, the semantic information is disregarded in this experiment. This gives a mapping process as shown in Figure 5. However, rather than providing the relation matrices of the graphs directly as input to the mapping nettwork, a two-phased mapping process is proposed. First, in phase one, a neural network compute *distributed representations* (Rumelhart et al. 1986) of the entire set of graphs. Second, in phase two, another neural network takes

as input the distributed representations and computes a correspondence matrix as output. This technique is somewhat similar to learning of familiy trees described by Hinton (1986). Note that since we want the neural network to find proper mappings on one set of graphs—that is, between one base graph and many analogue target graphs—we restrict the input data to target graphs only. Therefore the base graph G_b is not explicitly presented to the networks, but appears implicitly as a mapping in the correspondence matrices. A neural network trained by backpropagation (Ellingsen 1995) is used in both phase one and phase two.

Phase one: compute distributed representations

Given that each relation matrix $\mathbf{R}$ in the data set can be represented as a vector $\mathbf{x}$, the input data matrix to the neural network is defined as $\mathbf{X} = [\mathbf{x}_{t_1}, \mathbf{x}_{t_2}, \ldots, \mathbf{x}_{t_{37}}]^T$, where $\dim(\mathbf{x}_{t_i}) = 9$. The neural network process one vector $\mathbf{x}_{t_i}$ at a time.

The first phase is similar to an encoder network (Rumelhart et al. 1986), where a network learns the identity function $\mathbf{y} = \mathbf{F}(\mathbf{x}, \mathbf{w}_r)$, where $\mathbf{x}$ is the input vector, $\mathbf{y}$ is the approximated output vector, and $\mathbf{w}$ is the weight vector in the network. Our goal is to learn $\mathbf{w}$ such that $\mathbf{x} = \mathbf{y}$.

Note that in these experiments we have $\mathbf{F} : \mathbb{B}^k \ni \mathbf{x} \rightarrow \mathbf{y} \in \mathbb{B}^k$, where $\mathbb{B} = \{0, 1\}$, and $k = 9$ is the dimensionality of the data vectors. In a three-layer network, with one input, one output layer, and one hidden layer, if the network is able to approximate the identity function, the distributed representations for the graphs are given as activation values of the hidden layer units. Figure 15 shows a three layered fully connected 9–4–9 neural network, where, given an input vector $\mathbf{x} = (x_1, x_2, \ldots, x_9)$, approximates the identity function $\mathbf{F}$, where $\mathbf{y}$ is the approximated network output of the desired output $\mathbf{x}$. If the error signal, that is, the differences between the network output $\mathbf{y}$ and the input pattern $\mathbf{x}$, approaches zero, the activation values of the hidden layer units, given as vector $\mathbf{z} = (z_1, z_2, z_3, z_4)$, are the distributed representation for vector $\mathbf{x}$. The neural network shown in Figure 15 uses four units to represent the distributed representation of each input vector. Note that $\mathbf{z} \in \mathbb{R}^m$, where $m = 4$ is the dimensionality of the vector.

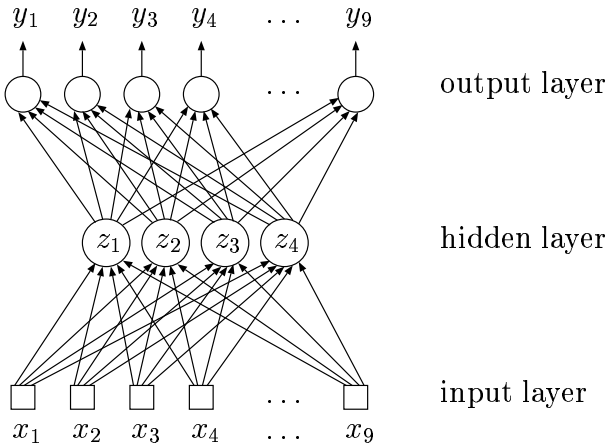

Figure 15: *A fully connected three-layered 9–4–9 neural network for learning the identity function* $\mathbf{F}$.

Given matrix $\mathbf{X}$ as input, the neural network was trained by backpropagation. The network terminated after 4169 iterations, whereupon the total mean square error was less that 0.003. Two learning rates were used in the network, one, denoted η, for the weights connecting the input to the hidden layer units, and one, denoted κ, for the weights connecting the hidden to the output units. The learning rates were set to low values, i.e., $\eta = 0.05$ and $\kappa = 0.15$, in order for the network to learn the identity function properly.

Finaly, the distributed representations for all input patterns is given by the matrix $\mathbf{Z} = [\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \ldots, \mathbf{z}_{t_{37}}]^T$, where $\dim(\mathbf{z}_{t_i}) = 4$. Note that $\mathbf{Z}$ is extracted from the activation values of hidden layer units in the very last repetition of the input data.

Phase two: mapping

In the second phase the distributed representations of the graphs, given as matrix $\mathbf{Z} = [\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \ldots, \mathbf{z}_{t_{37}}]$, are provided as input to the network, one vector $\mathbf{z}_{t_i}$ at a time. Given that each correspondence matrix $\mathbf{C}$ in the previous defined data set is represented as a vector $\mathbf{d}$, the desired output from the network, given input vector $\mathbf{z}_{t_i}$, is the vector $\mathbf{d}_{(b,t_i)}$. Thus, the entire data matrix for the desirede output is defined as matrix $\mathbf{D} = [\mathbf{d}_{(b,t_1)}, \mathbf{d}_{(b,t_2)}, \ldots, \mathbf{d}_{(b,t_{37})}]^T$, where $\dim(\mathbf{d}_{(b,t_i)}) = 9$.

The second phase can be defined as the mapping function $\mathbf{y} = \mathbf{M}(\mathbf{z}, \mathbf{w})$, where $\mathbf{z}$ is the input vector, $\mathbf{y}$ is the approximated output vector, and $\mathbf{w}$ is the weight vector in the network. Our goal is to learn $\mathbf{w}_m$ such that $\mathbf{d} = \mathbf{y}$. Thus, the function does a mapping in the domain $\mathbf{M} : \mathbb{R}^m \ni \mathbf{z} \rightarrow \mathbf{y} \in \mathbb{B}^k$, where $\mathbb{B} = \{0, 1\}$, $m = 4$ is the dimensionality of the input vector, and $k = 9$ is the dimensionality of the output vector.

Given the input matrix $\mathbf{Z}$ and the desired output matrix $\mathbf{D}$, two mutually disjoint data matrices are defined, one for learning and another for testing. The learning data matrices are defined as $\mathbf{Z}_l = [\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, \ldots, \mathbf{z}_{t_{20}}]^T$ and $\mathbf{D}_l = [\mathbf{d}_{(b,t_1)}, \mathbf{d}_{(b,t_2)}, \ldots, \mathbf{d}_{(b,t_{20})}]^T$, and the test data matrices are defined as $\mathbf{Z}_t = [\mathbf{z}_{t_{21}}, \mathbf{z}_{t_{22}}, \ldots, \mathbf{z}_{t_{37}}]^T$ and $\mathbf{D}_t = [\mathbf{d}_{(b,t_{21})}, \mathbf{d}_{(b,t_{22})}, \ldots, \mathbf{d}_{(b,t_{37})}]^T$.

The aim of this phase is not primarily to compute distributed representations, but to find an analogical mapping between a target and a base graph. To obtain this, a vector $\mathbf{z}$, representing the target graph, is provided as input to the network, and the objective is to compute an output vector $\mathbf{y}$, representing the appropriate analogical mapping between the target graph and a base graph. Figure 16 shows fully connected 4–6–9 neural network for approximating the mapping function $\mathbf{M}$, where vector $\mathbf{z} = (z_1, z_2, z_3, z_4)$ is the input and vector $\mathbf{y} = (y_1, y_2, \ldots, y_9)$ is the approximated output.

Given the data matrix $\mathbf{Z}_l$ and the desired output matrix $\mathbf{D}_l$ as input, the network terminated after 7036 iterations. The total mean square error was then less than 0.02. As with the network in phase one, two learning rates were used and set equally $\eta = 0.04$ and $\kappa = 0.04$.

Given the constraints on the correspondence matrix as discussed in Section 4.2, we obtain only 50 % correct mapping of the test data. In each output vector there are three individual mappings, one for each vertex from target to base. Hence, for all 17 test samples in $\mathbf{D}_t$ there are totally 51 mappings. The test results are reported

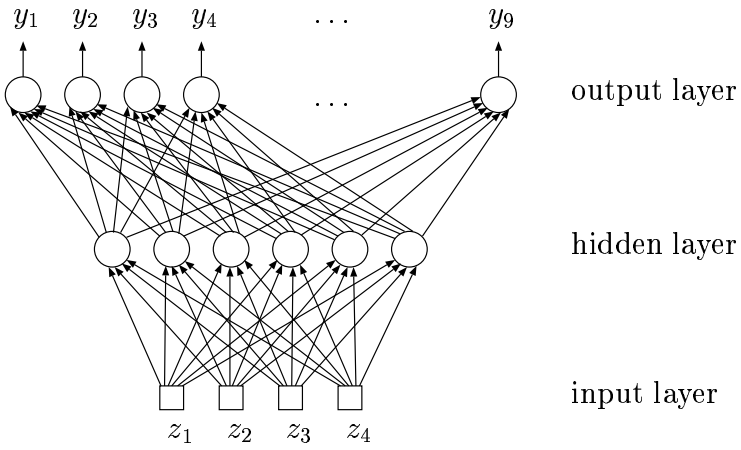

Figure 16: *A fully connected three-layered 4–6–9 neural network for learning the mapping function* **M**.

in Table 2, where the rows summarize the number of incorrect mapping of vertices over the entire test data. The 0 errors row denotes a correct mapping for all vertices between the target and base graphs, and the 3 errors row denotes that all the three vertices in the target graph is incorrect mapped to the base. The rightmost column shows the distribution of incorrect mappings in percent.

errors	*graphs*	*%*
0	1	6
1	4	24
2	9	53
3	3	17

Table 2: *Test results of the mapping experiments.*

As seen from the results in Table 2, only one target graph has correctly all three vertices correctly mapped to the base graph. A majority of the target graphs have two out of three vertices incorrectly mapped, and three target graphs have all three vertices incorrectly mapped.

5.3 Discussion

The modest results from this experiment may originate from an insufficient representation of the graphs in terms of relation matrices. The graphs contemplated here are subject for the variances discussed in Section 4.2. A situation may occur that two isomorphic graphs obtain different relation matrices simply because of different labeling. Further, two identical correspondence matrices does not necessarily define a mapping of identical pair of graphs. This makes the mapping task very difficult.

A hierarchical cluster analysis of the distributed representations in matrix **Z** computed in phase one is shown in Figure 17, and report similarities between the

representations according to the Euclidean distances. The cluster diagram is obtained from the square root of the squared Euclidean distances, based on the centroid method for clustering. Note that the diagram displays the clustering sequence and not the Euclidean distances among the clusters.

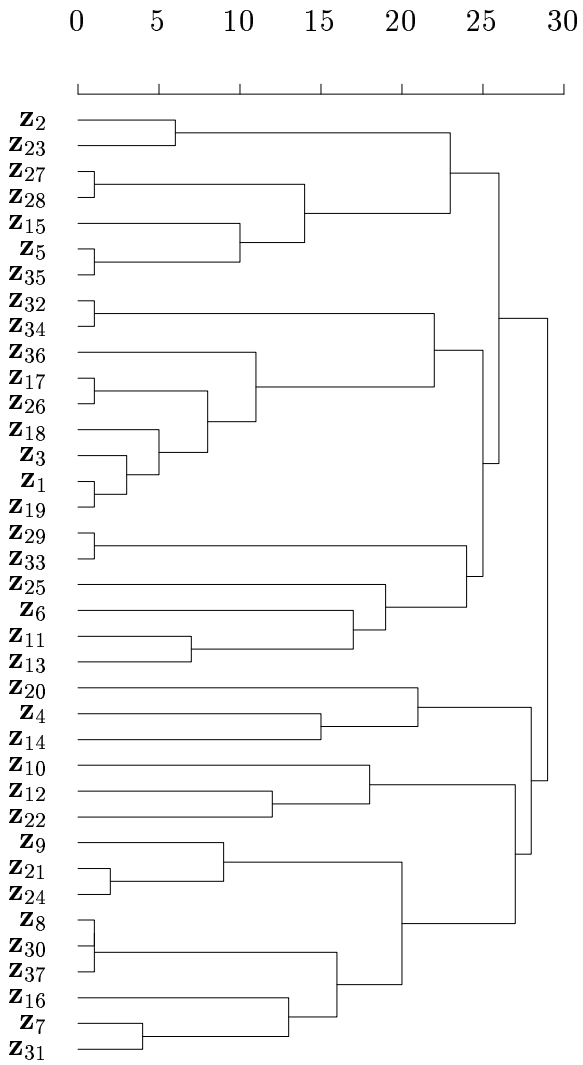

Figure 17: *Hierachical clustering of all the distributed representations computed in phase one.*

For example, the graphs $G_{t_{27}}$ and $G_{t_{28}}$, shown in Figure 18, have isomorphic structures and identical relation matrices. Therefore, their distributed representations are close to each other in the Euclidean space, denoted as $\mathbf{z}_{27}$ and $\mathbf{z}_{28}$ in the cluster diagram. The graphs G_{t_4} and $G_{t_{14}}$, also shown in Figure 18, do not have isomorphic structures or identical relations matrices. Hence, the distributed representations, denoted $\mathbf{z}_4$ and $\mathbf{z}_{14}$ in the cluster diagram, are not clustered close to each other.

The neural network in phase two of the mapping process accepted the distributed representations of the target graphs as input only. Although not discussed here, an experiment was defined where the relation matrices for both the base and the target

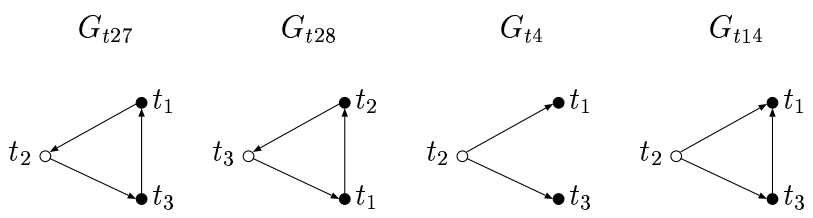

Figure 18: *Graphs from the input data, where $G_{t_{27}}$ and $G_{t_{28}}$ are isomorphic, whereas G_{t_4} and $G_{t_{14}}$ are not.*

graphs were provided as input to the neural network. The results for the network with two input matrices showed an even lower performance. Also, a one-phase mapping process was tested, where the relation matrices were provided directly as input to the neural network. This means that no distributed representations were explicit extracted for further processing, but computed by the very same network as learning the analogical mapping function. The results from these experiments did not report any promising results either.

A conclusion after analyzing the cluster diagram is that the network in phase one seeks to find distributed representations for the graphs based on similar relation matrices. The distributed representations can be regarded as features of the relations matrices, and the network seem able to extract common features for similar relation matrices. This is both good and bad news. First, this means that we cannot hope to represent the graphs directly as relation matrices simply because of they do not capture variances in the graphs. Second, if we can find an invariant representation of the graphs, we may assume that a neural network can extract the features for a subsequent mapping process. Given a technique for coding the graphs in an invariant way, the assumption is that similar graphs obtain similar distributed representations. Therefore, as discussed in Section 2.3, we can use a neural network to do similarity-based generalization and categorization due to formation of rich internal representations. However, an alternative coding scheme to the current relation matrices of the graphs is necessary in order to obtain better results.

6 Comparison with related work

An intrinsic feature of a distributed connectionist network is its ability to learn from a set of training samples and to generalize from these samples to produce an appropriate output. Likewise, in the ROSA project we want to learn a neural network from a set of graphs (representing the role models) and have the network to generalize from these graphs to produce an appropriate mapping. The difference between the proposed approach and ACME (Holyoak and Thagard 1989) and SME (Falkenhainer et al. 1989) is apparent by comparing the *representation* and *mapping* scheme.

6.1 Representation

The graphs of the role models represent input patterns to the neural network. From these patterns, the network can build generalized and reduced internal representations, which comprise the knowledge of the network. As the distributed representations capture common features of the patterns, we obtain a generalization capability where patterns are spread as activity patterns over the network's many units. The net effect is a similarity-based association where similar patterns obtain similar internal representations.

Compared to localist connectionist network, the roles and ports in the OOram role models correspond to units in the network, where inhibitory and excitatory links represent the strengths between the units. In ACME, the role models are presented to the network as sentences in predicate calculus. For example, given the two, yet simple, analogue and isomorphic role models shown in Figure 19, they can be represented to the ACME network as the enumerated propositions in predicate-calculus listed below the role models.

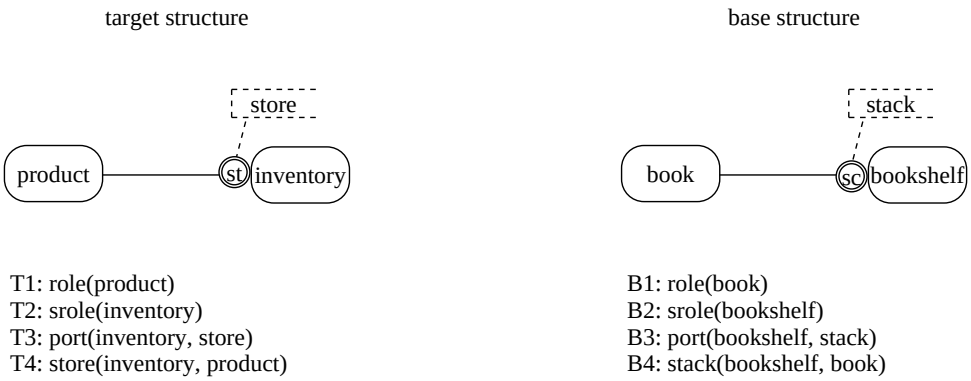

Figure 19: *Two simple analogue role models, one target and one base, including propositions for input to ACME. The proposition are sequentially enumerated.*

From these propositions, ACME builds a network of units linked by inhibitory and excitatory connections. The ACME network has one unit for each possible pairing (mapping) of the elements in the role models. There are also semantic units that connect elements with similar meaning, and pragmatic units that connect elements correlated with high-level plans and goals. An example of an ACME network[3] for the two role models in Figure 19 is shown in Figure 20.

The important thing is that ACME does not build an explicit internal representation of the role models. It builds, however, an explicit representation of possible mappings between elements of the target and base role models. Note that units for pragmatic centrality are not shown in the network in Figure 20. Symmetrical links are implemented to constrain the mapping, where solid lines denote excitatory connections and dotted lines denote inhibitory connections. An excitatory link between two elements strengthen a mapping, whereas an inhibitory link between two elements weaken a mapping. Since mapping between a target and a base role model

[3]The ACME network is adapted from Holyoak and Thagard (1989).

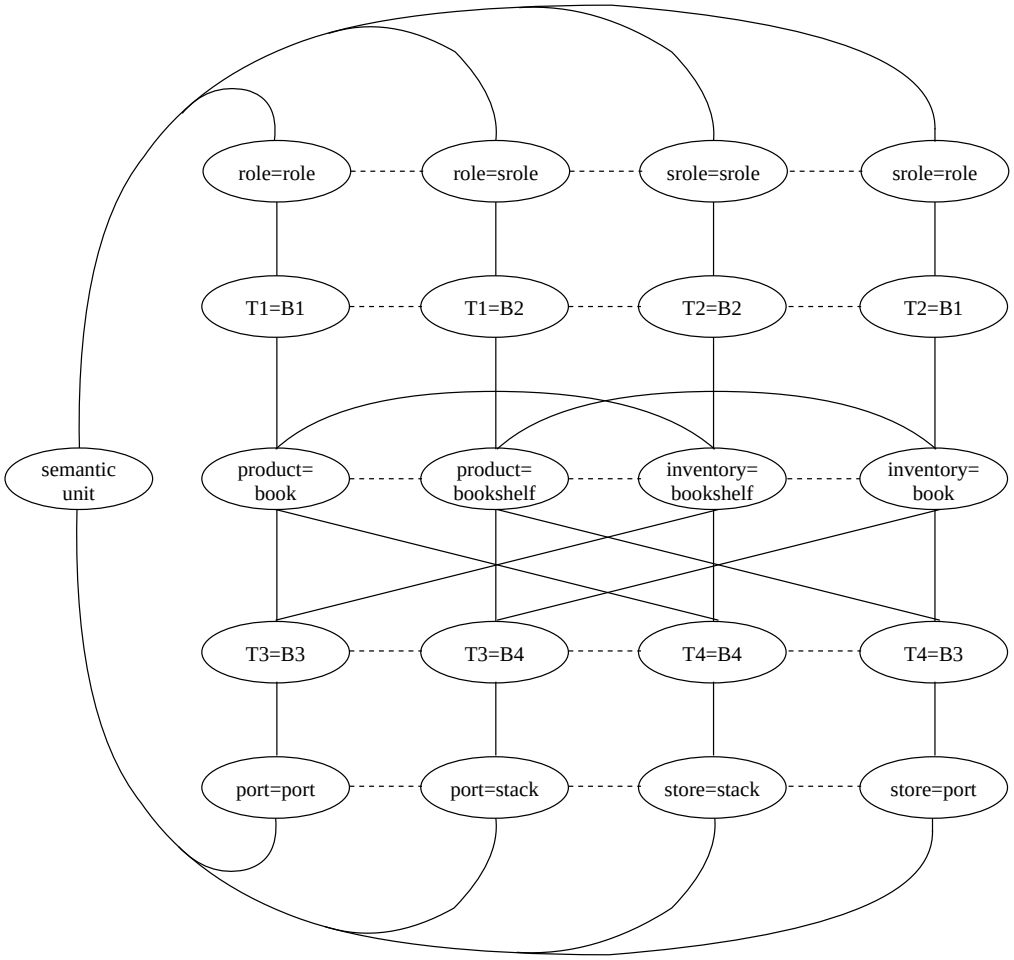

Figure 20: *ACME localist network for representing the role models in Figure 19.*

is explicitly representations in ACME, its ability to generalize over common features is rather poor. This is also the case for other symbolic-based analogical reasoning models, such as SME.

The distributed representations in the neural network are quite different from ACME and SME. A neural network does not clearly separate structural and semantic information as in the discrete models. A noticeable difference between a neural network and ACME/SME is to what degree the analogs and the mapping-constraints can be represented explicitly. ACME relies on a localist network where the units represent propositions and the connections represent constraints, whereas SME relies on matching rules. A neural network model has no similar explicit representation, where the analogs and mapping-constraints are spread over the many units in the network.

Thus, taking the relation matrices of the two role models as input to the neural network generates internal representations in a multi-dimensional real vector space. Since no particular unit in the neural network's internal representation denotes any aparticular element in the original structure, it is not obvious how a mapping is derived. A propitious strength of neural networks is their ability to extract features and do similarity-based generalization. This strength enable the neural network

to establish correspondences between similar internal representations representing similar elements in the original structures. However, a condition for a similarity-based mapping is the ability to discriminate among the elements in the structure. As demonstrated in the experiments in Section 5, treating the OOram role models as holistic patterns in terms of relation matrices have significant limitations.

Representation of analogs to the SME is similar to ACME, by sentences of predicate logic. Although there are some similarities between the two, such as mapping of higher-order relations in favor of isolated object descriptions, there are, nevertheless, fundamental differences. For example, SME does not build a network of units representing the potential mappings between the elements, but relies on rules to derive a global best mapping from a set of constituent hypotheses about element correspondences.

Both ACME and SME can handle semantic information represented as discrete and atomic elements. Rules for matching semantic similarities are provided in SME, e.g., *literal similarity* and *mere-appearance*, for comparing object descriptions. However, object descriptions are ignored unless they are part of some higher-order structure. ACME, on the other hand, exploits semantic information to a larger degree than SME by representing specific semantic units in the network with excitatory/inhibitory links for strengthening/weakening the potential mapping between the elements.

6.2 Mapping

The ACME builds, when presented with predicate logic propositions, a network of units that defines all possible pairings between the base and target elements. Essentially, compatible concepts are initiated by excitatory links, whereas incompatible concepts are initiated by inhibitory links. The initial network defines mapping-hypotheses that restrict the number of possible pairings. To arrive at a solution, the activation levels of the units in the network are updated repeatedly according to the activation value of the neighboring units and the connected links. The links between the units can be interpreted as constraints on the mapping, where ACME aims at satisfying as many of these as possible to arrive at a global best solution.

The SME also tries to derive at a best global mapping from a set of constituent hypotheses about element correspondences. SME begins the mapping process by computing local *match hypotheses* (conceptually similar to mapping units in ACME) guided by a set of rules. The match hypotheses are ordered according to their score. Subsequently, the local mathes are combined into maximal consistent collections of correspondences, called *gmaps*. ACME, on the other hand, seeks to find a single global best mapping, where all the constraints are treated in parallel. Moreover, SME prefers structural consistency, whereas ACME emphasize semantic similarity between elements as important for the mapping.

Comparing the mapping process of ACME and SME with the proposed neural network mapping model reveals some differences and similarities. First, ACME and the neural network-based mapping have some common feature in that both are based on a connectionist architecture. As such, both models have units and weighted connections that work in parallel, aiming at satisfying some constraints. However, neither the units nor the connections in the neural network have the same level of

abstraction as in ACME. This is due to the distributed representation in the neural network versus the localist representation in ACME.

Furthermore, the objective for the iterative learning of the ACME network is to find a best global mapping solution by satisfying as many constraints as possible. To find a best global solution is also the case for the neural network. However, the neural network aims at finding generalized internal representations of the inputs by grouping similar patterns into clusters. After a learning phase, a neural network can approximate an output for a novel input due to its ability to generalize. This is not the case in ACME, where every new pattern results in setting up a new network of the possible pairings between target and base elements. Both SME and ACME can dynamically build temporary structures in order to map analogs of different size and complexity. As discussed in Section 4.1, this is not easily obtain in neural networks (Barnden 1994b).

6.3 Why neural networks in the mapping of role models?

As thoroughly discussed in the previous sections, analogical mapping is acknowledged to be a structural problem. Therefore, a model of AR that emphasizes systematic relational structures, such as the SME, should be favored. However, there is a compelling reason for not using SME or ACME. The main argument is the limited ability for ACME and SME to *generalize* over a set of similar features. Essentially, we want the AR model to find similar representations for similar elements in the two analogs for a potential mapping. Thus, a mapping can be provided by establishing correspondences between elements with similar representations. This is a micro-feature-based mapping process that is not easily obtained in the symbolic-based AR where the number of possible mappings is scaled exponentially according to the increased complexity of the analogs. In a neural network the input patterns are distributed among the units, and where a learning algorithm finds the fine-grained internal representations in a self-organized manner. Thus, the representation of the structures (graphs) is "flattened" in the network, which imply that there is no need for complex search algorithms along the structures.

6.4 Alternative coding schemes

In the current coding scheme the graphs are represented *holistically* to the neural network as relation matrices. However, as acknowledged in Section 4.1, the graphs have compositional structures that are sensitive to variances in the representations. It might be more appropriate to use a representational scheme that conveys the structure-sensitivity of the graphs. Fodor and Pylyshyn (1988) argues that neural networks are essentially inadequate for supporting representations that are sensitive to compositional structures. This means that for a system to be structural sensitive, the representations must be decomposed into their constituent parts by using some destructing operation, yet preserving the integrity of the original structure. Typical examples of this are natural language processing and grammar parsing, often represented symbolically as structured lists and trees. However, Fodor and Pylyshyn's critique has been demonstrated to be insufficient, and their conclusion is refuted by Chalmers (1990). Several authors have shown that distributed connectionist net-

works can represent and process compositional information. For example Pollack (1990) demonstrates how the RAAM (Recursive Auto-Associative Memory) model can be used to discover compact distributed representations of symbolic compositional structures, such as lists and trees. The RAAM builds fixed-sized reduced representations of variable-sized symbolic sequences and trees. An extension of RAAM, called Labeling RAAM (LRAAM), is proposed by Sperduti (1993) for encoding of arbitrary labeled structures, such as directed garphs. Similar ideas for representing complex compositional structures are tensor products (Smolensky 1990) and HRRs (Holographic Reduced Representations) (Plate 1994).

7 Conclusions

The aim of this paper was to provide a representational scheme for a connectionist based analogical mapping of role models. A major goal has been to provide a formal representation of the role models used in the mapping process of the ROSA project, which conforms to the distributed paradigm of connectionist data processing. The role models were represented as directed graphs and placed in relation matrices. Both semantic and structural information can be extracted from the initial retrieval phase of ROSA, and by combining the two types of information, we believe a more effective analogical mapping can be obtained. The output of the network is a mapping scheme that establishes correspondences between roles and their relations from a target to a base model.

A two-phased mapping process was proposed and tested on randomly generated graphs. In the first phase the graphs representing the role models were given a distributed representation by a auto-associative backpropagation network. In the second phase the distributed representation were provided as input to another backpropagation network, whose objective was to compute an output that represents the appropriate analogical mapping from a target graph to base graph. Results from the experiments indicate that the proposed approach is limited because the graphs are presented as "holistic" patterns. A consequence of this is that the network cannot discriminate among the constituents of the structures.

Comparison with related work shows that distributed connectionist networks are not much explored in analogical reasoning. Although there is reported some work on connectionist-based AR, most models are nevertheless based on localist and marker-passing connectionist networks. Previous study also indicates that neural networks are difficult to use in analogical reasoning due to its reliance on a distributed representation of the knowledge. In most problem domains in analogical reasoning, and artificial intelligence in general, the knowledge is symbolic and has a compositional structure.

In order to obtain better results a coding of the graphs that facilitate discrimination of the constituents of the structures seems required. Further work includes investigation and experimentation with different types of distributed connectionist networks, particular with a more powerful representation in mind. Also, methods for combining structural and semantic information are important in that respect. Alternative representational schemes of the role models, such as RAAM (Pollack 1990) and LRAAM (Sperduti 1993) seems promising in that respect.

References

Barnden, J. A. (1994a). Introduction: Problems for high-level connectionisms. In J. A. Barnden and J. B. Pollcak (Eds.), *High-Level Connectionist Models*, Volume 1 of *Advances in connectionist and neural computation theory*, Chapter 5, pp. 1–16. Norwood, New Jersey: Ablex Publishing Corporation.

Barnden, J. A. (1994b). On the connectionist implementation of analogy and working memory matching. In J. A. Barnden and K. J. Holyoak (Eds.), *Analogy, Metaphor, and Reminding*, Volume 3 of *Advances in connectionist and neural computation theory*, Chapter 8, pp. 327–374. Norwood, New Jersey: Ablex Publishing Corporation.

Biggerstaff, T. and C. Richter (1989). Reusability framework, assessment, and directions. In T. J. Biggerstaff and A. J. Perlis (Eds.), *Software Reusability. Vol. I. Concepts and Models*, Chapter 1, pp. 1–17. Addison Wesley.

Bjørnestad, S., B. Tessem, K. Tornes, and G. Steine-Eriksen (1994). Useful characteristics for identifying analogous specifications. Technical Report No. 24, ISSN 0803–6489, Dept. of Information Science, University of Bergen.

Chalmers, D. J. (1990). Why fodor and pylyshyn were wrong: The simplest refutation. In *Proceedings of the Twelfth Annual Conference on the Cognitive Science Society*, Hillsdale, NJ, pp. 340–347. Lawrence Erlbaum Associates.

Dinsmore, J. (Ed.) (1992a). *The Symbolic and Connectionist Paradigms: Closing the Gap*. Hillsdale, New Jersey: Lawrence Erlbaum Associates.

Dinsmore, J. (1992b). Thunder in the gap. See Dinsmore (1992a), Chapter 1, pp. 1–23.

Eksridge, T. C. (1994). A hybrid model of continuous analogical reasoning. In K. J. Holyoak and J. A. Barnden (Eds.), *Analogical Connections*, Volume 2 of *Advances in connectionist and neural computation theory*, Chapter 4, pp. 207–246. Norwood, New Jersey: Ablex Publishing Corporation.

Ellingsen, B. K. (1995, June). An object-oriented approach to neural networks. Technical Report No. 45, ISSN 0803–6489, UIB-IFI.

Falkenhainer, B., K. D. Forbus, and D. Gentner (1989). The structure-mapping engine: Algorithm and examples. *Artificial Intelligence 41*, 1–63.

Fichman, R. G. and C. F. Kemerer (1992, October). Object-oriented and conventional analysis and design method. *IEEE COMPUTER 25*(10), 22–39.

Fodor, J. A. and Z. W. Pylyshyn (1988). Connectionism and cognitive architecture: A critical analysis. *Cognition 28*, 3–71.

Fugini, M. G., O. Nierstrasz, and B. Pernici (1992, August). Application development through reuse: the Ithaca tools environment. *ACM SIGOIS Bulletin 13*(2), 38–47.

Gentner, D. (1983). Structure mapping: A theoretical framework for analogy. *Cognitive Science 7*(2), 155–170.

Haugeland, J. (1986). *Artificial Intelligence: The Very Idea*. MIT, Massachusetts: A Bradford Book.

Hinton, G. E. (1986). Learning distributed representations of concepts. In *Proceedings of the Eigth Annual Conference of the Cognitive Science Society*, pp. 1–12. Hillsdale, NJ: Lawrence Erlbaum Associates.

Holyoak, K. J., L. R. Novick, and E. R. Melz (1994). Component processes in analogical transfer: Mapping, pattern completition, and adaption. In K. J. Holyoak and J. A. Barnden (Eds.), *Analogical Connections*, Volume 2 of *Advances in connectionist and neural computation theory*, Chapter 2, pp. 113–180. Norwood, New Jersey: Ablex Publishing Corporation.

Holyoak, K. J. and P. Thagard (1989). Analogical mapping by constraint satisfaction. *Cognitive Science 13*, 295–355.

Johnson, R. E. and B. Foote (1988, June/July). Designing reusable classes. *Journal of Object-Oriented Programming 1*(2), 22–35.

Kokinov, B. N. (1994). A hybrid model of reasoning by analogy. In K. J. Holyoak and J. A. Barnden (Eds.), *Analogical Connections*, Volume 2 of *Advances in connectionist and neural computation theory*, Chapter 5, pp. 247–318. Norwood, New Jersey: Ablex Publishing Corporation.

Korson, T. and J. D. McGregor (1990). Understanding object-oriented: A unifying paradigm. *Communications of the ACM 33*(9), 40–60.

Lange, T. E. (1992). Hybrid connectionis models: Temporary bridges over the gap between the symbolic and the sybsymbolic. See Dinsmore (1992a), Chapter 10, pp. 237–289.

Minsky, M. (1985). A framework for representing knowledge. In R. J. Brachman and H. J. Levesque (Eds.), *Readings in Knowledge Representation*, pp. 246–262. Morgan Kaufmann.

Morel, J.-M. and J. Faget (1993, March 24–26). The REBOOT environment. In R. Prieto-Díaz and W. B. Frakes (Eds.), *Advances in Software Reuse, 2nd International Workshop on Software Reusability*, Lucca, Italy, pp. 80–88. IEEE CS Press.

Newell, A. and H. A. Simon (1972). *Human Problem Solving*. Englewood Cliffs, NJ: Prentice-Hall.

Owen, S. (1990). *Analogy for Automated Reasoning*. London: Academic Press.

Plate, T. A. (1994). *Distributed Representations and Nested Compositional Structure*. Ph. D. thesis, University of Toronto.

Pollack, J. B. (1990, nov). Recursive distributed representations. *Artificial Intelligence 46*(1-2), 77–105.

Reenskaug, T. and E. Andersen (1992). OORASS: seamless support for the creation and maintenance of object-oriented systems. *Journal of Object-Oriented Programming 5*(6), 27–41.

Reenskaug, T., P. Wold, and O. A. Lehne (1996). *Working With Objects. The OOram Software Engineering Method*. Manning Publications Co.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning internal representations by error propagation. See Rumelhart and McClelland (1986b), Chapter 8, pp. 318–352.

Rumelhart, D. E. and J. L. McClelland (1986a). *Parallel Distributed Processing: explorations in the microstructure of cognition*, Volume II: Psychological and Biological Models of *Computational models of cognition and perception*. MIT Press, Mass.

Rumelhart, D. E. and J. L. McClelland (1986b). *Parallel Distributed Processing: explorations in the microstructure of cognition*, Volume I: Foundations of *Computational models of cognition and perception*. MIT Press, Mass.

Schank, R. C. (1980). Language and memory. *Cognitive Science 4*, 243–284.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton, New Jersey: Princeton University Press.

Smolensky, P. (1990). Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence 46*(1-2), 159–216.

Sperduti, A. (1993, May). Labeling RAAM. Technical Report TR-93-029, International Computer Science Institute, Berkeley, CA.

Tessem, B. (1995a, March). Analogy for software modeling. Technical Report No. 41, ISSN 0803–6489, UIB-IFI.

Tessem, B. (1995b). Analogy for software modeling: The base filtering problem. In *Norsk informatikkonferanse '95*, pp. 83–94.

Tessem, B. (1996, October). Using structure abstractions in retrieval for analogical reasoning. Technical Report No. 51, ISSN 0803–6489, UIB-IFI.

Tessem, B., S. Bjørnestad, K. Tornes, and G. Steine-Eriksen (1994). ROSA = Reuse of Object-oriented Specifications through Analogy: A project framework. Technical Report No. 16, ISSN 0803–6489, Dept. of Information Science, University of Bergen.

Thagard, P. (1988). Dimensions of analogy. In D. Helman (Ed.), *Analogical Reasoning*, pp. 105–124. Kluwer Academic Publishers.