SUBMISSION DRAFT for BBS. Date: November 10, 2000.

Paper Title (*not finalized yet. Please vote or suggest*):

Building large-scale hierarchical models of the world with binary sparse distributed representations.

or

Building the world model with binary sparse distributed representations.

Dmitri A. Rachkovskij

V. M. Glushkov Cybernetics Center
Pr. Acad. Glushkova 40
Kiev 03680
Ukraine

dar@infrm.kiev.ua
(*Preferable contact method*)

Ernst M. Kussul

Centro de Instrumentos
Universidad Nacional
Autonoma de Mexico
Apartado Postal 70186
04510 Mexico D.F.
Mexico
ekussul@servidor.unam.mx

Keywords:
analogy,
analogical mapping.
analogical retrieval,
APNN,
associative-projective neural networks,
binary coding,
binding,
categories,
chunking,
compositional distributed representations,
concepts,
concept hierarchy,
connectionist symbol processing,
context-dependent thinning,
distributed memory,
distributed representations,
Hebb,
long-term memory,
nested representations,
neural assemblies,
part-whole hierarchy,
representation of structure,
sparse coding,
taxonomy hierarchy,
thinning,
working memory,
world model.

Rationale for soliciting Commentary

Many researchers agree on the basic architecture of the "world model" where knowledge about the world required for organization of agent's intelligent behavior is represented. However, most proposals on possible implementation of such a model are far from being plausible both from computational and neurobiological points of view.

Implementation ideas based on distributed connectionist representations offer a huge information capacity and flexibility of similarity representation. They also allow a distributed neural network memory to be used that provides an excellent storage capacity for sparse patterns and naturally forms generalization (or concept, or taxonomy) hierarchy using the Hebbian learning rule. However, for a long time distributed representations suffered from the "superposition catastrophe" that did not allow nested part-whole (or compositional) hierarchies to be handled. Besides, statistical nature of distributed representations demands their high dimensionality and a lot of memory, even for small tasks.

Local representations are vivid, pictorial and easily interpretable, allow for an easy manual construction of both types of hierarchies and an economical computer simulation of toy tasks. The problems of local representations show up with scaling to the real world models. Such models include an enormous number of associated items that are met in various contexts and situations, comprise parts of other items, form multitude intersecting multilevel part-whole hierarchies, belong to various category-based hierarchies with fuzzy boundaries formed naturally in interaction with environment. It appears that using local representations in such models becomes less economical than distributed ones, and it is unclear how to solve their inherent problems under reasonable requirements imposed on memory size and speed (e.g., at the level of mammals' brain).

We discuss the architecture of Associative-Projective Neural Networks (APNNs) that is based on binary sparse distributed representations of fixed dimensionality for items of various complexity and generality. Such representations are rather neurobiologically plausible, however we consider that the main biologically relevant feature of APNNs is their promise for scaling up to the full-sized adequate model of the real world, the feature that is lacked by implementations of other schemes (proposals).

As in other schemes of compositional distributed representations, such as HRRs of Plate and BSC of Kanerva, an on-the-fly binding procedure is proposed for APNNs. It overcomes the superposition catastrophe, permitting the order and grouping of component items in structures to be represented. The APNN representations allow a simple estimation of structures' similarity (such as analogical episodes), as well as finding various kinds of associations based on context-dependent similarity of these representations. Structured distributed auto-associative neural network of feedback type is used as long-term memory, wherein representations of models organized into both types of hierarchy are built. Examples of schematic APNN architectures and processes for recognition, prediction, reaction, analogical reasoning, and other tasks required for functioning of an intelligent system, as well as APNN implementations, are considered.

ABSTRACT (medium - 250 words)

Many researchers agree on the basic architecture of the "world model" where knowledge about the world required for organization of agent's intelligent behavior is represented. However, most proposals on possible implementation of such a model are far from being plausible both from computational and neurobiological points of view.

Implementation ideas based on distributed connectionist representations offer a huge information capacity, flexibility of similarity representation, and possibility to use a distributed neural network memory. However, for a long time distributed representations suffered from the "superposition catastrophe". Local representations are vivid, pictorial and easily interpretable, allow for an easy manual construction of hierarchical structures and an economical computer simulation of toy tasks. The problems of local representations show up with scaling to the real world models, and it is unclear how to solve them under reasonable requirements imposed on memory size and speed.

We discuss the architecture of Associative-Projective Neural Networks (APNNs) that is based on binary sparse distributed representations of fixed dimensionality for items of various complexity and generality, and provides a promise for scaling up to the full-sized model of the real world. An on-the-fly binding procedure proposed for APNNs overcomes the superposition catastrophe, permitting representation of the order and grouping of structure components. These representations allow a simple estimation of structures' similarity, as well as finding various kinds of associations based on their context-dependent similarity. Structured distributed auto-associative neural network is used as long-term memory, wherein representations of items organized into part-whole (compositional) and concept (generalization) hierarchies are built. Examples of schematic APNN architectures and processes for recognition, prediction, reaction, analogical reasoning, and other tasks required for functioning of an intelligent system, as well as APNN implementations, are considered.

ABSTRACT (small - 100 words)

Most proposals on possible implementation of the real world model required for organization of agent's intelligent behavior are far from being plausible both from computational and neurobiological points of view. We discuss the architecture of Associative-Projective Neural Networks (APNNs) that is based on binary sparse distributed representations of fixed dimensionality for items of various complexity and generality, and provides a promise for scaling up to the full-sized model of the real world. Structured distributed auto-associative neural network is used as long-term memory, wherein representations of items organized into part-whole (compositional) and concept (generalization) hierarchies are built. Examples of schematic APNN architectures and processes for recognition, prediction, reaction, analogical reasoning, etc., as well as APNN implementations, are considered.

1. Introduction

Research in artificial intelligence (AI) and cognitive science suggests that organization of intelligent behavior requires the world model - a system of knowledge about domain and about intelligent system itself. The world model must allow the intelligent system (hereafter referred to as the System) (the agent?) to recognize the world states and their sequences perceived with sensors, and to respond to them according to the System's goals, e.g., by a sequence of actions. In so doing, efficient performance of the System requires predictions of the environment and the System changes, including the results of their interactions.

Following the principle of system and environment adequacy (e.g., Antomonov 1974; 1969; see also Ashby 1956), it may be inferred that complexity and organization of the world model must be adequate to complexity and organization of environment and of the System's behavior. Of special interest are the principles of construction, implementation, and operation of adequate world models at the complexity level of humans and mammals - the only known examples of intelligent systems capable of efficient functioning in complex real environment.

1.1. Organization of the world model

Various principles of adequate real-world model organization were proposed by many authors from diverse branches of science - from philosophy and cognitive science to AI and system analysis. A picture that emerges from the works of Hebb (1949), Amosov (1967), Quillian (1968), Rumelhart, Lindsay, & Norman (1972), Anderson & Bower (1973), Minsky (1975), Schank & Abelson (1977), Anderson & Hinton (1981), Rumelhart, Smolensky, McClelland, & Hinton (1986), Booch (1994), Barsalou (1999), Nirenburg &Raskin.(2001), and many other researchers may be sketched as follows (see also section 2).

The world model comprises interrelated models of various items (objects and attributes) stored in the System's memory. The objects may be real and abstract, e.g., physical bodies, processes, situations, their parts, concepts, sentences, etc. The objects are characterized by permanency of some attributes, dynamic or static (properties, parts, relations with other objects), or by regularities of their changes. However, division of items into objects and attributes is rather relative. The item considered as an attribute in one case is considered as an object in another case. An object may be a part, and therefore an attribute, of a more complex object. An attribute-property may be composite, and therefore may have its own attributes. However, there probably exist elementary, terminal attributes that do not have their own attributes.

Models are associated with each other. For example, a model of an object is linked with the models of its attributes and with the models of more complex objects which have it as an attribute. Thus, models have hierarchical compositional structure. It is a part-whole or a compositional hierarchy. Therefore, the models of attributes may be considered as component models connected with each other in the model of the whole object. A model-whole of some hierarchical level can be a model-part of models of more complex objects.

A model-whole is built up using the models-components - attributes of various modalities and structural parts with their relationships. It concerns various objects, such as usual things, visual scenes, action plans, or sentences. The models of a ball or a dog in visual modality of the System have component models - the attributes of shape, color, texture, size. Besides, a dog model has its component models in structural modality - the models of body, head, paws, tail. The structure models also include component models of spatial relations between body-part models, and their changes in the process of dog movement.

The model of event "approach, dog, ball" includes the models of objects-parts (dog, ball) and the model of changing their spatial relation (approach). The model of situation "approach, dog, ball; bite, dog, ball; burst, ball" includes the models-parts of three events "approach, dog, ball", "bite, dog, ball", "burst, ball".

In addition to the "part-whole" relations, another type of relations exist between the models. It is a "class-instance", "type-token", "concept-object", "is-a", "hyponymy" relation. Existence of class models causes perception of classes and concepts. Classes are groups of objects with common attribute sets. Classes make up a classification hierarchy - another major type of hierarchy. There are more abstract and more specific classes. The model of a more general class includes less number of models of characteristic attributes. If a specific class is a member of a more general class, the former shares the attributes of the latter and also has certain attributes of its own.

For example, there exists the model of Rover (a specific dog). Rover is a spaniel. A spaniel is a dog. There exist models of "spaniel" and "dog" representing more and more general classes with nested attributes. The permanency of attribute combinations of specific objects and their classes allows knowledge accumulated as a result of the System's experience to be exploited and transferred to new objects, similar but non-identical to the known ones.

Associations or links should exist between the models of classes of various degree of generality. A model of a more specific concept has an access to a more general class model and to its models-attributes, allowing attribute inheritance and transitions between the models of various generality levels. Models of various generality are also linked by part-whole associations.

Associations between the component models of a model-whole allow transitions between the models, such as models of objects, classes, and their attributes, permitting the System to perform recognition, predictions, and reactions. Perception of certain combination of models-attributes allows recognition of an object or a class. For example, perception of a rolling colored sphere and furry barking quadruped running after it permits the objects belonging to the classes of balls and dogs to be identified.

Identification of a model by observable attributes, e.g., visual appearance, provides an access to other, unobservable attributes. Recognition of a situation allows the System to get access to the attributes associated with the evolution of similar situations in the past, and to predict its outcomes. The models of System's actions must be formed and associated with the models of objects and situations on the basis of its previous activity. This allows the System to respond to various objects and situations by actuating appropriate programs of behavior consisting of action models that also have a hierarchical structure. By recognition of a dog and a rolling ball, the System may anticipate that the dog will run after, bite it, and the ball may burst. If the ball is valuable for the System, it may bring into operation the model of some complex action to save the ball, e.g., to reach the ball before the dog, or distract the dog from the ball.

Thus, the large-scale model of the real world requires building, storage, modification, and operation of a very large number of models of various complexity and generality (objects, situations, actions, their classes, attributes, associations), organized in the "part-whole" and "is-a" hierarchies, that are mainly formed as a result of unsupervised learning.

1.2. Implementation of the world model

Representation of the models and associations defines largely the implementation of mechanisms of learning and operating with the models. In turn, the way of the world model implementation determines its adequacy and scaling potential, as well as peculiarities and efficiency of the System functioning.

However, most approaches to implementation of the world models (schemata, scripts, plans, frames, semantic networks, memory organization packages, conceptual dependencies, etc.) are neither neurologically relevant, nor they even in principle allow scaling up to the size and complexity of the real world in respect to memory, speed, variety and flexibility required. Most of those approaches are based on traditional symbolic AI representations that are close to localist connectionist models in neural network implementation.

In local schemes, representation and storage of each model requires a separate memory location. Associations between models are implemented explicitly, by physical connections or pointers. Representation of a model-whole is an isolated combination of connections or pointers to the component models, therefore it does not contain immediate information about the content of the component models. Finding similarity between composite models demands complex processes of their decoding through the elementary component models and finding the correspondences between the component models of various hierarchical levels.

Such peculiarities of local representations lead to a number of drawbacks in respect to building of a full-scale model of the real world. They include a weak potential for scaling in terms of required memory size; difficulties of representation of similar objects and attributes and estimation of similarity; problems with generalization, storage of classes and class instances, emergence of classification (is-a) hierarchy. It should be noted, that these drawbacks are not very important for flat, unstructured models (Page 2001), but manifest themselves in full measure for nested composite models (see section 3).

Distributed representations offer a potential to overcome these drawbacks. This potential is mainly connected with an excellent information capacity, natural representation and estimation of similarity, possibility to use distributed memory that allows storage and generalization processes to be simplified. The attractiveness of distributed representations was emphasized by the paradigm of cell

assemblies (Hebb 1949) that influenced the work of Marr (1969), Willshaw (1981), Anderson (1972), Nakano (1972), Grossberg (1971), Kohonen (1972), Palm (1980), Hinton, McClelland & Rumelhart (1986), Kanerva (1988), and many others. However, until recently the ideas inherent in Hebb's paradigm have not been developed enough, especially the problem of representation of part-whole hierarchies in the structure of assemblies (e.g,. Legendy 1970; von der Malsburg 1986; Feldman 1989, Milner 1996).

1.3. Associative-Projective Neural Networks

Our approach to building a rich, well-scaling, hierarchical world model is developing in the framework of the paradigm of Associative-Projective Neural Networks (APNNs) (e.g., Kussul 1992; Kussul, Rachkovskij, & Baidyk 1991). The APNNs are a neural network architecture based on structured compositional distributed representations. It is one branch of work on modeling of thinking processes initiated by N.M. Amosov in early 60-s in the V.M. Glushkov Institute of Cybernetics, Kiev, Ukraine (Amosov 1967). Another branch of that work consisted in building localist semantic networks - M-networks (Amosov, Kasatkin, Kasatkina, & Talayev 1973). Originally these two directions were followed in parallel, resulting in the world's first autonomous robot (vehicle) controlled by neural networks in natural environment (Amosov, Kasatkin, & Kasatkina 1975; Amosov, Kussul, & Fomenko 1975).

The foundations of the APNN paradigm were proposed by Kussul in 1983. In the framework of APNNs, an attempt was made to combine the hierarchical organization of the world model of Amosov and other researchers with the advantages of Hebb's assemblies and distributed representations. The employment of distributed representations having a high information capacity allows the world's diversity to be reflected in APNNs. The use of binary sparse representations causes the simplicity of implementations and is not without neurobiological relevancy.

Distributed representations of models-wholes in APNNs are bound and reduced distributed representations of their component models (structural parts and other attributes). The procedure of Context-Dependent Thinning is used for their construction (see section 6). Using such representations, it is easy to represent similar items and find similarity of composite models-wholes immediately, without decoding and mapping of their component models.

In APNNs, the links between associated models-wholes and models-parts, as well as between models of classes and instances, are not explicit. They are found by the context-dependent similarity of corresponding distributed representations. For example, similarity of representations of models-wholes and component models allows finding the component models by the model-wholes, and the model-whole by the component models. Multi-level character of part-whole hierarchy is reflected in multiple levels of APNNs. Using of distributed associative memory for storage of model representations maintains a high storage capacity, creates a natural basis for classification hierarchy, provides an easy finding of similar representations necessary for recognition, classification, prediction, association, etc.

1.4. An outline of the article

In this article we consider the basic ideas and architectural solutions underlying the paradigm of APNNs. In section 2 general problems concerning hierarchical organization of the world model are discussed. In section 3 a comparative analysis of local and distributed representations for implementation of a large-scale compositional model of the world is provided. In sections 4 a basic structure of APNNs is described. In section 5 we consider Hebb's concept of cell assemblies, its development, using of an assembly neural network as a distributed associative memory in APNNs, and emergence of concepts and category-based hierarchy therein. In section 6 the Context-Dependent Thinning procedure that is used for binding and normalization in APNNs is given. In section 7 the methods and examples of representation and processing of composite structures in hierarchical APNNs are presented. In section 8 representation and processing of complex sequences in APNNs and some related application tasks are discussed. Software and hardware implementations of APNNs are considered in section 9. Sections 10-11 are discussion and conclusion respectively.

2. Organization of the world model

Organization of adequate world model must reflect perceived organization of the world, and also provide execution of processes required for the System's behavior - learning, recognition, prediction, reaction, etc. In this section we outline the views we share with many researchers concerned with these problems in various fields - from philosophy and cognitive science to AI and object-oriented design. Approaches to neural network implementation of the world model will be considered in the remainder of this article.

2.1. Perceived organization of the world

2.1.1. Objects and attributes

Information about states of the external and internal world arriving to the brain from sensors of various kind is diverse and variable. However, our perception of the world partitions it into objects that interact, change, have some characteristics, reflecting availability of regularities in the organization of the world and human capability to extract and use them. The nature of those regularities is diverse - some are determined by physical or chemical laws, other result from the regularities in behavior of living creatures, etc. Thus, we perceive the regular organization of the world in terms of objects and attributes.

It is assumed that the System must extract some characteristics that permit objects to be separated, identified, and distinguished. These characteristics are called attributes or features. Objects are characterized by permanency of some combinations of attributes or regularities in their changes.

For example, physical bodies and their aggregates are spatially structured and bounded tangible entities. Permanency of attributes corresponding to those characteristics provides the basis for perception of physical bodies. A table, a tree, fog, a manufacturing plant are examples of bodies. In addition to physical bodies, there are objects of various kind, real and abstract, that are still characterized by permanency of some attributes, e.g., a chemical process, a word, a sentence, a model, a centaur.

Attributes may have various nature, modality, complexity, etc., providing a basis for their classification into various groups or categories - rather arbitrary, though. Attributes may be static (states) or dynamic (actions, processes, events). For example, the attributes of spatial location of a body at some time instant are its state, and their changes with time are some actions or events. Processes and actions have a phase structure, consisting of sequences of state changes.

Some attributes are inherent in a thing independently of other things. Those are properties or intrinsic attributes of a thing. Some properties are immediately extracted from the information perceived by sensors of various modalities, such as green, round, large, loud, musk, soft, hot, sweet, etc. Sensors inside the body give the attributes of its state, e.g., proprioceptive sensors provide position of body parts, chemical sensors provide the attributes of hunger, thirst, pleasure, emotions, etc. Extraction of more abstract properties, such as kind, beautiful, brave, clever, probably requires more complex processing of information about items and their relationships.

Division of entities into objects and attributes is rather arbitrary, because entities that manifest themselves as attributes may appear in other relations as having attributes, i.e., as objects. Entities indivisible into attributes will be referred to as elementary, primitive, or base-level attributes (green, warm).

Entities (or items) do not exist in isolation. There exist numerous relationships between them - links, interactions, dependencies. These are spatial (above), temporal (after), causal, part-whole relations, etc. Relations are attributes that require several objects to manifest themselves. They may be directed (A above B, A sold B a book) or undirected (A and B are neighbors). Arguments of directed relations have roles (agent A, object B, etc.).

2.1.2. Part-whole relations and hierarchy

A part-whole relation is global and universal. It penetrates the world at any scale level, from micro to macro, from elementary particles to galaxies. It spans the objects of all complexity levels and domains, real and abstract, organizing them into diverse multi-level hierarchies of parts-wholes, components-composites. This type of hierarchy is known as "part-whole", "part-of", "meronymy-holonymy", "aggregation", "structural", "modular" or "compositional" hierarchy. Complex objects at one

hierarchical level are composed of inter-related objects-parts of the lower level of part-whole hierarchy. Parts of objects, in their turn, may be composite objects, comprising the parts of lower hierarchical levels.

Parts may be found in physical bodies and other objects. For example, a proton is a part of an atom, an arm is a part of a body. A process consists of phases or operations, a motor act consists of elementary movements. A relational instance consists of predicate and arguments. "Bite", "dog", "ball" are parts of situation. "Bite, dog, ball", "burst, ball" are parts of cause-effect relation. Object parts may be considered as its attributes. For example, a proton is an attribute of an atom, a head is an attribute of a body, "bite", "dog", "ball" are attributes of the corresponding situation.

Part-whole hierarchy reflects the hierarchy of interactions, interdependencies of things at various levels of the integral world. Objects interact, interrelate with each other to a varying degree. Some combinations of objects interact stronger than others, making up stable structures - complex objects-wholes consisting of parts.

Steadiness, regularity, stability of object-whole structure may be different. Parts of some objects are able to exist only as constituents (only in composition) of the whole. Parts of a live organism can only exist together, providing each other with necessary resources and supporting integrity of the organism. Other objects-wholes have a less stable structure than physical bodies. For example, parts of situations are not so mandatory, can exist without each other and be involved in many situations.

2.1.3. Object structure and global attributes

It is known from the theory of systems that functions of a system can not be reduced to the functions of its components (emergent behavior). For hierarchical systems, function of a system at certain hierarchical level is determined by its structure - the components of previous hierarchical levels, their interactions and functions. Variability of system structure and component functions causes variability of system functions (Antomonov 1974).

If an object-whole is considered as a system, and its global attributes (properties, functions, behavior) are considered as the system functions, then the global attributes are determined by the objects-parts and their interactions (though do not reduce to them). Therefore, the structure of objects contains important information for the intelligent System.

For example, green color of a forest observed from a distance is specified by green color of tree crowns, and, in turn, it is specified by green color of leaves. If it is raining and we recognized a tree from a distance by its integral appearance, we may suppose it can protect us from rain. However, we have to choose the tree with a thick and branchy crown. Also, the type of associative memory in a neural network and its characteristics depend on the characteristics of neurons, the structure of connections, learning rules.

2.1.4. Contexts

The object structure - parts and their relationships - contains an important information about its global attributes. Therefore a known structure of an object provides a bottom-up context that allows properties, functions, behavior of the object to be predicted or specified. For example, behavior of a lion without a mane (lioness or cub) or with a wounded paw will differ from the behavior of lion with another body structure.

Objects-parts are constituents of various objects-wholes (bodies, scenes, situations). The attributes of parts may depend on the whole that provides a top-down context. For example, behavior of a lion depends on whether a prey or another lion is in front of it.

Besides, global attributes of an object may depend on each other. This type of context may be named "side context", or "same-level context". For example, hunger or satiety influences the lion's behavior.

2.1.5. Class-instance relations and category-based (generalization) hierarchy

Despite diversity of the world, there exist similarity between very different objects. Observing the objects, discovering their similarities and differences, we extract essential and stable combinations of attributes. The common character of objects in terms of their constant combinations of attributes allows

their organization into classes - groups or sets of objects sharing common patterns of attributes, such as sensor features, parts, properties, behaviors, etc.

This allows accumulated experience to be transferred to new objects and situations and predictions to be made. For example, similar looking objects behave similar. Assigning an object to a class, we may assume that it inherits essential attributes typical of (inherent in) that class. If the mechanism of classes was absent, knowledge about each novel object would be gained from scratch.

Particular object is considered as an instance of a class. It has its own specific attributes in addition to common attributes of the class. Whereas an object is a concrete entity that exists in the world, a class represents only an abstraction. Probably, it stems from existence of class models in brain (see section 2.2.2).

Classes make up hierarchies. Some classes may possess common attributes allowing their integration into a more general or abstract class. Those classes have less number of common attributes and combine less similar objects. The levels of this hierarchy are the levels of generality-specificity (generalization-specification), or the levels of abstraction of categories or concepts, from very specific to very general.

For example, let us consider the taxonomy of plants. Trees, bushes, grass are plants. Birches, poplars, pines are trees. Birches, poplars are broad-leaved trees. Specific birch is an instance of birches.

This type of hierarchy allows the information about the classes of objects to be represented economically, without replicas for each object, but storing it in the model of common class. It also allows making categorical inferences and predicting the object attributes, once an object is classified.

How such hierarchies are formed and represented by humans is unknown. Probably the following basic schemes are realized.

- Supervised schemes. The teacher postulates the combination of attributes that assigns an object to a class. Or, the teacher points to the objects belonging a class, and their common combinations of attributes are revealed. Not all objects of a class share the same combinations of attributes.
- Unsupervised schemes. Objects are observed and manipulated. Common combinations of attributes are extracted and form classes. Or, some "prototype" - a typical class representative - is formed, and sufficient resemblance of an object to the prototype determines the class membership.

2.2. Hierarchical structure of the models

2.2.1. Compositional structure of the world model

The world is structured. It may be supposed that discreet perception of the world through various objects and attributed results from existence of their models in the brain. Therefore, the System must have internal representations - the models of various kinds of attributes - properties, relations, actions, parts, etc. There must be also models of various objects - simple and complex, real and abstract - bodies, situations, scenes, processes, etc., including the System's model.

The world is integral. Integrity of the world in its partitions, perception of objects of varied complexity as integral items characterized by combinations of interconnected attributes and existing in interaction and interdependence with each other, suggests compositional structure of models and existence of associations or interconnections between the models.

It is presumed that the model of an object-whole is a composite model in which an aggregate of component models is interconnected or associated. The component models may be the models of an object's global attributes, or the models of an object's structure - objects-parts and their interrelations. Also, the model of object-whole may be a model-component of more complex objects or relations. Interconnections must fix the structure of component models in the composite models. Also, they must provide an access or association between the models.

For example, in the model of a physical object, the models of its attributes of various sensory modalities should be interconnected, as well as its structure model (if known) and the models of behavior - various actions in which the object may be involved. In the model of a situation, the models of its components are associated - the models of objects and their relations, the models of its possible evolution - the models of changes of the objects' states and interactions, the models of results of those interactions; as well as the models of assessment of those results' usefulness or harm for the System, and the models of the System reactions - if the System is one of the objects.

Compositional structure of models is hierarchic. Models-wholes of some hierarchical level may be models-parts of higher level models. A model-part of some model-whole may have a complex compositional structure of its models-components of lower hierarchical levels. Such an organization of models might be a basis for reflection of this type of hierarchical relations in human consciousness.

Interconnections between component models and composite models provide mental associations between parts and wholes. In thinking, it is reflected in associations between objects, between attributes, between objects and attributes - one of the items induces the thought about another one. Existence of models-wholes in which the models-parts are interconnected serves as a basis for perception of the world in terms of objects. Existence of models-parts influences perception of not only structured physical objects, but also various abstract objects, processes, plans, etc.

Connections between the models of both the same and different hierarchical levels also permit operations or processes necessary for the System functioning to be performed (section 2.3). Apparently, models, connections, and processes operating on them can be implemented by various ways. The implementation issues will be considered in further sections.

2.2.2. Category-based structure of the world model

The world is diverse. Therefore, there must exist an abundance of models of various modalities and degree of complexity - of attributes, relations, actions, objects, as well as situations where the objects and the System find themselves. Thus, it is necessary to form, associate, store, modify a tremendous number of models.

The world is regular. Despite a multitude of models, a number of component models and their combinations occur together in various composite models. The combinations of component models typical for some group of composite models and interconnected with each other make up models of classes or categories. In these terms, concept is a named class model, that is, a class model that has its name as one of the component models.

The number of attribute models in the class model is less than that in a specific object's model. However, those attributes are essential, typical for the class.

The occurrence of class models allows a model of new, unknown object to be constructed not "from zero", but to use class models as the basis. It facilitates the construction process, making it supplementing the class model with the individual attributes - object peculiarities. In this way knowledge accumulated in the System is extended to new objects.

Classification or type-token relations or associations - object X belongs to class Y - are established in consciousness, reflecting the commonality of attributes. An interconnection between the model of a class and the model of a class instance (a specific object) must correspond to this relation.

Thus, similarity of objects in their diversity provides the basis for another kind of hierarchical organization of models, that is usually referred to as the hierarchy of general-specific, the hierarchy of similarities, kind-of, or classification hierarchy. The degree of similarity may be different, and the classes are ordered in the hierarchy by the degree of generality. A class with a greater set of attributes is included in a class with a smaller set of attributes. Therefore, class models must be connected with each other, so that the model of a less general class must have access to the model of a more general class and its models-attributes.

2.2.3. Plurality of hierarchies

A model-attribute may be a component of various models-objects, and a model-object may be a part of various models of more complex objects, e.g., scenes. Various partitions of an object model into models-parts are also possible. Depending on the context, the System may extract different attributes from perceived information, or use various subsets of attributes to describe a certain object. Thus, a model may be a part of many part-whole hierarchies.

For example, a tree may be described from various standpoints.

(a) If a tree is described with respect to its functioning as a living organism, then such parts as roots, a trunk with branches, leaves may be selected. Roots are responsible for absorbing water and minerals from the soil. The trunk and branches transport raw materials up to the leaves. The leaves use the substances from the roots and CO_2 from the air to produce organic substances through photosynthesis. They are distributed to the trunk, branches, and roots for their growth. If the property of trees to grow vertically is of interest, than the attribute of roots to fix the trunk in the ground and the attribute of trunk to support the crown may be of interest.

(b) Probably, the sensory model-whole of a tree in visual modality may include visual models of its parts - a crown and a trunk, and spatial relations "crown above trunk", "trunk sticks up the ground". The visual model of the crown may incorporate such component models-attributes as "leaf texture", "green color", "crown shape". This visual model, as well as models of crown from other

modalities, such as the model of "rustle" attribute from acoustical modality and "leaf touch" attribute of tactile modality, are component models of the composite model - multi-modal sensory model of the crown.

(c) In the model of a crown, its component models, such as models of branches and leaves, may also be distinguished, and may be decomposed into still smaller parts-models. For example, the property of a crown as a whole to protect from the sun and rain is influenced by the size and shape of leaves.

Classification is fuzzy and flexible as well. An object may belong to many classification hierarchies, in a similar manner as it can belong to various part-whole hierarchies. Classification useful for the System depends on the context, the goals, etc. This specifies the class model used by the System, and therefore the set of attribute models to which access is gained. That is, an object may inherit from different classes, depending on the context. For example, a tree may be considered as a shelter, as a kind of fuel, as a source of food, as a kind of plant, etc. Moreover, the set of attributes used to assign an object to a certain class is not fixed. Various object models of the same class may have different sets of attributes common with the models of the class attributes.

Unlike combining parts into wholes, where the whole requires simultaneous availability of several parts, a concept of any subclass belongs to the more abstract class combining those subclasses. Therefore, the category-based hierarchy may be considered as an OR-type hierarchy, and the part-whole hierarchy may be considered as an AND-type hierarchy.

Not only instances, but also classes may form part-whole hierarchies. For example, this poplar in the yard has the short crooked trunk and the large asymmetric crown. But, poplars has long straight trunks and symmetric pointed crowns. Also, trees have trunks and crowns.

Models-wholes coinciding with models of classes, and their models-parts coinciding with models of subclasses are also possible. For example, a wardrobe, a bed, a table are kinds of furniture and parts of furniture suit.

Hierarchical levels may be considered as levels of abstraction. In the part-whole hierarchy, these are levels of scale increasing-decreasing. Abstractions of the higher level relate to diminished items, wherein the details of their parts are indistinguishable, and abstractions of the lower level are enlarged items with more details. At each scale level, details essential for that specific level of consideration are "visible". In the category-based hierarchy, these are levels of generalization-specification. Abstractions of the higher level (class) are more general, containing most important attributes, and abstractions of the lower level (subclass) are more specified, containing more attributes, but both refer to items of a certain scale level.

2.3. Manipulations with the models

We have considered the structural organization of models. Some peculiarities of model formation will be discussed in section 2.4. In this section, let us consider processes operating on the formed models.

Models and their interconnections are stored in long-term memory. The basic type of models' processing may be seen as their associating - finding in long-term memory and extracting to working memory the model most fitting the input model(s) (probe) in the current context. The retrieved model is exploited for the System's needs - e.g., extracting other models, their detailed comparing, actuating the System's effectors, etc.

Three basic types of associations between the models may be distinguished. They are all context-dependent and include finding the model of the same complexity level most similar to the probe, finding the component models of the probe, and finding super models of which the probe is the component. These three types of associations may be accomplished between the models of various levels of generality - specific objects or concepts.

As shown in section 3 and further, these processes may be implemented very differently, depending on the representation of models and their interconnections.

2.3.1. Similarity of models and object identification

The value or measure of similarity and difference between composite models is a function of their common and different attribute models, as well as of similarity degree of corresponding (but non-identical) attribute models. This value depends on the context which can determine the set of attribute models in the composite models selected for comparison, and importance of those attributes.

Let us consider how an object can be recognized by its observable sensory attributes, that is, how its model is identified in the System's memory. The model-whole of a perceived object is formed

in working memory from the models-components of observable attributes. The context may influence selection of models-attributes forming the object's model. Then the similarity of this probe model to the models from long-term memory is determined. The model from long-term memory similar enough to the probe model is selected as the recognition result or as a hypothesis about the perceived object. For example, a ball can be recognized by the attributes round, orange, smooth, elastic, or an orange can be recognized by the attributes round, orange, smooth, sweet.

Observable attributes of various modalities can be used to find the similarity of models. They can be static and dynamic sensory attributes characterizing appearance and behavior of an object. They can also be object's parts and their relations, in statics and in dynamics. Finding similarity and difference of complex composite models requires taking into account similarity of their component models - parts, their relationships, roles of the parts. Since the size of long-term memory for the real-world model is huge, implementation of model representations and similarity estimation procedures must be computationally efficient. It must allow a parallel process of finding similarity of nested composite model-probe to all structured models in long-term memory, or at least suggest good hypotheses about the most similar models, that could be then verified thoroughly.

To identify the class model of a probe, coincidence of some subset of observable attributes with the attributes essential or typical for the class and contained in its model is enough. Observed attributes may not fully coincide with the models of specific objects, if the observed object is unknown to the System or perceived in an unknown context or from a new viewpoint.

2.3.2. Associations between models, predictions and responses

After getting assess to some model, e.g., by some set of its component models, the System must get access to other components of the model. The System must also get access to composite models that have this model as their component. Such an access or association is accomplished via the interconnections between component and composite models.

For example, after recognition of an object by selection of its model using observed attributes of certain modalities, the System must get access to unobserved attributes, or to the component models of other modalities. Unobserved attributes could be the attributes of visual appearance, structure, past behavior, experience of interaction with the System, the System's responses. This allows the System to predict the structure and behavior of objects or evolution of situations and form its own behavior.

If an object is recognized by some observable parts and their spatial relations, prediction of availability and position of unobservable parts can be made, facilitating their search and recognition. If the head and the tail of a dog jut out of grass, it may be predicted that the body and the legs are between them in the grass.

The choice of particular component models predicting behavior of the recognized object depends on the context - other objects and their behavior, time and space conditions, current state of the object, etc. For example, prediction of the lion's behavior depends on whether it hunts or sleeps, hungry or thirsty, presence of prey, its proximity, etc. (see also section 2.1.4).

The models associated with the recognized object must also contain information concerning experience of various versions of System-object interactions, their results and assessment. This provides information for action planning, allowing the System to choose or construct the plan of adequate interaction with the object in particular situation (see also section 2.5). Once the model-program of behavior (of interaction with the object) is chosen, the System must get access to its components - models of actions. Models of actions also have a hierarchical structure, their components of the bottom level control the System's effectors.

The number of predictable attributes in a class model is less than in a model of specific object, but those attributes are more reliable for making predictions about an unknown object. Therefore, classification allows making more justified predictions than recognition of the closest model of specific object which would transfer too many attributes without sufficient reasons.

However, the confidence of predictions for unknown objects is lower than for familiar objects. Predictions and hypotheses should be verified and refined with caution. When encountered a dog, it is better to suppose that it can bite. Then one could discover with joy that the small spaniel Rover living next door usually does not bite. However, premature generalization can spoil this joy by bites from spaniel Fido and from even smaller, but spiteful Pekinese Dolly.

2.4. Model formation and modification

The world is changeable and the System is adaptive. Therefore, to a large extent the models and their connections must be formed and modified in interaction with environment. This concerns formation of object models, their classes, non-inborn attributes. This also concerns organization of models into part-whole and categorization hierarchies.

Learning requires the availability of modifiable long-term memory in the System. For operations with the models, there must exist working memory that can promptly change its content interacting with environment and long-term memory.

The process of learning must be substantially unsupervised. The System does not know in advance which combinations of attributes correspond to objects of various complexity and their classes and what responses are appropriate to observed objects. Therefore the System must accumulate some kind of statistics of observed attributes, extract their regular combinations, and use this information and its variations for formation of models. This also concerns the models of the System's interactions with objects.

The teacher facilitates and accelerates the learning process because it allows a substantial reduction of statistics extraction stage for model formation. The use of words is especially efficient because it allows description of situations that were not met by the System, introduction of concepts, etc.

Let us consider some peculiarities of thinking that must influence the model formation.

2.4.1 Chunking

Practice shows that an individual can not comprehend, recognize, operate a large number of different objects simultaneously. It seems that the maximum number of chunks of information that a person can simultaneously comprehend is on the order of 7+-2 or even 4+-1 (Cowan 2001). This size is probably connected with the span of human short-term or immediate memory or attention focus. It does not depend on the information content of a chunk (e.g., 4 letters or 4 syllables or 4 words).

Such peculiarities of perception may be connected with those of model structure and operation, which in their turn may be connected with the peculiarities of the world organization. The probability of stable combinations of parts is higher if their number is lower. However, if an object-part has appeared that can make up objects-wholes by regular combinations of such parts, then it can build up large objects. Examples are atomic and molecular structure of things, cell structure of living matter. Regularities and dependencies of attributes are also easier to discover if their number is modest.

However, we are able to perceive complex objects by appropriate organization of information referred to as chunking. In the process of chunking, groups of small number of parts combine into the whole. Chunking is recursive; chunks of a certain hierarchical level (parts) are combined into the chunk of the higher hierarchical level (whole). In so doing, we can comprehend a complex object level by level, focusing sequentially on its parts of different levels, and at each level we need only comprehend a few parts at once. This is possible for a human, because the complexity of information chunks is not important.

Thus it is assumed that in forming part-whole hierarchical organization of models, the limitation of chunking is taken into account. The model-whole at each level and at any moment is formed from just a few models-parts, and only a few component models can be associated with the model-whole simultaneously. Moreover, it seems we can understand only those objects or systems, for which we have such a hierarchical structure of models.

In various contexts, the components of a chunk of a certain model may be different. For example, when we look at a car from outside, we pay attention to its shape, color, wheels; from inside we look at the windows, seats' upholstery; when driving, we pay attention to the controls. Therefore, the total number of models-parts in the model-whole may be much more than the number of models in a chunk, since the model-whole may be formed sequentially from many chunks of few components each (see also section 2.1.3, 2.1.4, 2.2.3).

2.4.2. Model synthesis and abstraction

Let us consider synthesis of models-wholes from models-parts. As this takes place, a small combination of component models must be selected from their total number, probably with the focus of attention (voluntary or involuntary), and integrated into the model-whole in working memory.

Formation of a stable model-whole of an object or a class of objects in long-term memory is probably facilitated by a regular joint occurrence of certain component models. However, models may

be combined in the human brain not only reflecting the perceived world or previous experience, but also in a novel way, as evidenced by creativity.

When sets of component models are combined into a composite model, coarsening, idealization, and abstraction of a composite object take place. The object becomes coarsen, it is considered as a simpler one. Only a small subset of component models and their interactions essential in particular context are selected for inclusion into the composite model. The autonomy of those components and importance of those connections is idealized. Abstraction from the internal structure of components and connections is performed. The degree of this abstraction is different for various implementations of models and can exert a determining influence on the System functioning, as shown in section 3.

A relative autonomy of models also allows focusing on a single part - an object of some level of part-whole hierarchy, its components and functions, ignoring the others. Thus the levels of part-whole hierarchy represent different levels of abstraction, each built upon the other, and each understandable by itself. We choose a given level of abstraction to suit our particular needs. For example, computer repairing at the level of PCB modules does not require knowledge of chips and all the more a gate-level design of chips and an atomic design of gates.

2.4.3. Model analysis and decomposition

In order to perceive an object as a whole, there must exist its model as a whole. The models of its global attributes must be connected in this model. However, if the System has no information about the object structure, there are no connections between the model-whole and the models of attributes-parts, and probably no models of such parts at all. Besides, not all possible global attributes of an object may be connected in the existing model-whole.

Probably, for each object of unknown structure, the System must seek to find out in the composition of the object its structural components and their interrelations and construct their models if they do not exist. This also concerns the models of its global attributes. Elucidation or refinement of the object's attributes may be realized by inspection of the object, that manifests itself in humans and mammals in the feeling of curiosity. This can also be done by analogy to the available models of similar objects with known structure.

The found component models of the object must be connected with each other and with the model-whole in the process of model synthesis. Such a combination of analysis (extraction of parts from the whole) and synthesis (incorporating the component models into the model-whole) enrich the model-whole, because the models-parts become its members.

Thus, the information in the models concerning the object's structure allows the System (see also section 2.1.3):

- to recognize or facilitate recognition of an object, especially a class instance;
- to predict the whole object by its parts and the parts by the whole, e.g., in sensory perception or in design of an object with desired global attributes (behavior, functions);
- to interact with the object-whole through the proper interaction with its parts;
- to solve large problems or design large system. The object must be decomposed into smaller and smaller parts, each of which may be solved or refined independently, and which will jointly produce the sought-for result (see also object-oriented design approach, e.g., Booch 1994);
- to transfer knowledge by analogy. The objects comprising similar parts in similar relations are due to have similar appearance and behavior (attributes).

2.5 Functional acts as constituents of intelligent behavior

Amosov (1979, 1967) considers activities of the System as an aggregate of "functional acts" aimed at the achievement of some goals. The goals are the models of desired states of environment and the System which are set externally or by the System.

Following Amosov, the scheme of a simplified functional act (FA) may look as follows. At the first stage of FA, the System perceives the states or actions of the world. Internal representations of their attributes form temporal models in short-term or working memory. The temporal models are recognized by a flexible comparison with the object models in long-term memory. As a result, a "secondary model" is obtained - the world picture rewritten in terms of the internal models. The process of recognition is hierarchical: temporal and recognized models may be transferred to higher hierarchical levels, where more complex models are recognized. After recognition of the model of

current situation, the System can extract associated experience of evolution of similar situations in the past, and hence predict future of the current situation and its results for the System, with due regard to the System's own actions.

At the next stage, situation estimation, action planning, and decision making are performed. The recognized situation and its forecast is estimated in respect to its potential for meeting the current System's needs. If the estimate is high enough, the model of target state of environment is created in accordance to the System's goal and current state.

It is assumed that the component models of "current state" - "action" - "resulting state" are associated and stored in the System's memory. Therefore, the action model that transforms current situation to target situation can be found provided with those states. The object may be the System itself, and the action may be the System's action.

The actions might be not only elementary, but generalized ones - e.g., some behavioral programs. The model of generalized action is unfolded into the plan - the sequence of more specific actions towards the model of target situation. Since the action models include the states of the System and environment resulting from each action, as well as the sensible cost of achieving the result, the change of state resulting from a certain action can be predicted. The action plan corresponding to the optimal total sensible evaluation is selected. If the target can be achieved with reasonable costs, a decision is made concerning the plan realization, followed by the realization stage itself.

The plans and actions have a hierarchical structure. The plan of a boy for getting to school does not consist of detailed actions, such as movements of arms or legs. Instead, major parts of plan exist, such as "exit the house", "go to the bus stop", "wait for the bus", "take the bus", etc. Each part of the plan has each own subgoals. To exit the house, it is necessary to go out of the flat, to come downstairs, to go into the street, etc. At some level of detailed elaboration of the plan, its fulfillment is realized by a hierarchical sequence of elementary motor acts. For example, the System effectors can be actuated to effect the environment or to search for an additional sensory information.

The feedback from the real results achieved and efforts spent is used to correct the plan of current actions. Besides, the experience obtained is used for modification of action, result, and cost models in memory.

Behavior is built as an aggregate of FA. The larger FA of higher hierarchical levels are unfolded through the smaller acts of the lower levels, and the latter, in their turn, are realized through concrete actions. Besides, existence of various goals of the System leads to the necessity of a complicated interaction between the *network* of FA targeted at the goals' accomplishment.

2.6. Two types of hierarchy: a local summary

The part-whole hierarchical organization of the models allows
- knowledge about the world - the objects and attributes of various complexity and their relationships - to be represented, stored, and processed;
- a hierarchical, discrete and continuous structure of the world to be reflected;
- operation with many objects despite a limited span of short-term memory by hierarchical chunking of parts into wholes;
- variation of the scale of consideration, by focusing on the model objects of certain complexity or level in the part-whole hierarchy;
- associations between the models of parts and wholes, thus enabling
- identification or recognition of observed objects;
- evaluation of similarity and differences between objects;
- prediction of unobserved attributes and objects, such as appearance and behavior of objects, evolution of situations, etc.;
- facilitation of object recognition and search;
- retrieval of or construction of the models of interactions with the objects;
- realization of interactions with the objects through effectors;
- storage of information on how the structure (parts and relations) influences the properties of the object-whole, how the attributes influence each other, and using this information for interactions with objects through their parts, for creation of new objects with predicted properties and behavior, etc.

The category-based (or generalization-specification) hierarchical organization of the models allows
- classification of known and unknown objects and thus extends to them knowledge accumulated in the System;

- an economical storage of knowledge about objects in the models of their classes, without its replication in the model of each specific object;
- a facilitated construction of new models for which the classes (or categories) are given;
- a variable level of abstraction by focusing on the objects or concepts of certain level in the category-based hierarchy;
- facilitation of learning, especially with the teacher, by permitting transfer of generalized knowledge to the System.

3. Distributed vs local representation of the models

Let us consider how the models of items of various complexity and generality (diverse objects and attributes, their classes, etc.) could be implemented in connectionist networks.

A connectionist network is a brain-like medium consisting of interconnected units or nodes - analogs of neurons or their distinct populations. A unit has a level of activation that is projected from it to connected units along weighted connections. The weights are analogs of synaptic efficacies. The level of activation of a unit is a function of the sum of weighted activations of other units connected to it.

Two basic types of information representation in connectionist networks are local and distributed one. We will consider construction and processing mechanisms, and their peculiarities, for both representation types. The implementation issues to be addressed include: how to represent models of elementary and composite items (objects and attributes), and part-whole associations between component and composite models; how to store and retrieve such representations of various hierarchical levels using associations between the item models of the same or different hierarchical levels; how to represent, store, and process items and classes of different generality and their hierarchical relations.

To simplify the exposition and discussion of the associated problems, we will use toy implementation examples. They reveal some basic problems and approaches to their solution for both representation schemes. Obviously, the implementation schemes for particular applications should have their own peculiarities.

We are mainly concerned with hierarchically organized items and their models, according to the framework presented in the previous section. Examples of this section are mainly unimodal and do not take into account the sequence of components in composite items. These generalizations will be considered in sections 7 and 8. Besides, let us consider representations that use binary units. For distributed representations, the outline of this section is oriented on the mechanisms developed in the framework of APNNs. These mechanisms will be discussed in more details in the following sections. However, most of the arguments and examples are applicable to the other schemes of distributed representations of compositional structures, such as HRRs of Plate (1991, 1995, 2000) or BSC of Kanerva (1994, 1996, 1998).

3.1. General definitions

General definitions of distributed and local representations may be as follows. In distributed representations, any item can be represented by a distributed pattern of activity over its pool of units. In local representations, any item can be represented by a single active unit from the pool (Figure 3.1).

"Any item" means an item of any nature and complexity level. It may be a model of an elementary attribute, a relation, a physical object, a scene, a class of thereof, etc. "Can be represented" leaves the mechanisms for allocation or assignment of units to represent a particular item for further specification. However, after the mechanism has been established, "with a local representation, activity in individual units can be interpreted directly" (Thorpe 1995, p.550). Local representation of an item is close to the notion of symbol in symbolic representations.

"A distributed pattern" means that many units from the pool participate in the representation of an item, and that a single unit participate in representation of many items. Therefore, "with distributed coding individual units cannot be interpreted without knowing the state of other units in the network" (Thorpe 1995, p.550).

A pattern of activity over a pool of N units can be considered as a codevector, where each element represents the state of the corresponding unit. Bold font will be used to denote codevectors, and italic font - for the corresponding items. For binary units, the codevectors are binary.

3.2. Elementary items

Elementary (indecomposable) items correspond to the models that have no internal structure (their component models are unknown or unimportant). Possible examples may be such properties as color, weight, speed, labels, probably some kinds of relations, etc.

3.2.1. Representation

In *local* representations, an elementary item is represented by a single unit. An active state of that unit corresponds to the presence of the item. Single elementary item is encoded by a codevector with a single 1 (Figure 3.1a). In *distributed* representations, different codevectors with a number of 1s are used to encode different items (Figure 3.1b).

3.2.2. Information capacity

In *local* representations, diversity of possible representations is limited by the number of units in the pool. The pool of *N* units is capable to represent locally up to *N* items. In *distributed* representations, codevectors overlap. Therefore much more than *N* items can be represented by different binary codevectors from the pool (up to (*M* from *N*), where *M* is the number of 1s in a codevector).

3.2.1. Similarity

Similarity of items encoded by codevectors may be characterized by two parameters: the composition (content) and the value (degree). The content of similarity is the elementwise product of the corresponding codevectors.
For binary codevectors, it is the overlapping 1s:

$$\boldsymbol{O(X,Y)} = \boldsymbol{X}\&\boldsymbol{Y}. \quad (3.1)$$

The degree of similarity is estimated by dot product of those codevectors. For binary codevectors, dot product is simply the fraction of coinciding 1s. Normalized similarity value of **X** to **Y** is

$$S(\boldsymbol{X},\boldsymbol{Y}) = |\boldsymbol{X}\&\boldsymbol{Y}|/(|\boldsymbol{X}| + small\_const), \quad (3.2)$$

where $|\boldsymbol{X}|$ is the number of 1s in $\boldsymbol{X}$, & is bitwise conjunction. *small_const* is used for codevectors of different density (see also Carpenter & Grossberg 1987; Marshall 1990) and will usually be omitted hereafter for simplicity.

In local representations, since the codevectors are orthogonal, similarity of elementary items is "all-or-none". If two items are represented by the same unit, they are identical. If they are represented by different units, they are non-similar.

In binary distributed representations, the 1s in codevectors overlap. So, the degree of similarity of two items can be expressed by the degree of correlation of their codevectors. The same codevector is used to represent identical items. To represent non-similar items, randomly generated codevectors are usually used.

Dot product of randomly generated codevectors is approximately the same for large N, and its mean value and standard deviation depend on the statistical distribution of the codevector elements. An average overlap of two random binary codevectors is $S=M*M/N*N$. For dense binary random codevectors ($M=0.5N$), $S=0.25$. For sparse binary random codevectors, e.g., $M<0.01N$, $S<0.0001$. The codevectors with larger overlap are used to represent similar items. Such sparse codevectors are deemed to be used in APNNs (see section 5). In Figure 3.1b, ***A*** and ***B*** are correlated, ***A*** and ***C*** are uncorrelated.

3.3 Simple composite items

Simple composite items are the items whose models have elementary models (of attributes and/or parts) as their components.

3.3.1. Fully-articulated representation - representation by superposition

In order to represent a simple composite item, superimposed representations of its component items can be used. For superposition of binary codevectors, let us use their bitwise disjunction. Superposition may be also considered as the result of simultaneous activation of component representations. The result is the codevector of the same dimensionality, where the 1s correspond to the 1s of the component codevectors (Figure 3.2). Note that for local representations the result of superposition can not be considered as truly local. We will call such a representation "full" (fully-articulated) or disjunctive. Below, "reduced" - pure local and distributed - representations of composite items will be considered.

3.3.2 Similarity

Similarity content for *local* representations is defined by identical components (similarity of the component sets). In *distributed* representations, the content of similarity is defined not only by the similarity of component sets, but by the similarity of representations of components as well. For example, in local representations, if there are two similar components, *A* and *A'*, represented by different units, they will not contribute to similarity. In distributed representations, *A* and *A'* with common 1s in their codevectors will contribute to the similarity of the composite representations where they are the components.

Both for local and distributed representations, the value of similarity of two simple composite items represented in the fully-articulated manner can be determined as in Eq. 3.2. For example, let ***X*=*AvBvC, Y*=*AvBvD***, where v denotes superposition (disjunction). Both for local representations and for binary distributed representations with a small overlap of component codevectors, the value of similarity will be approximately 2/3 (neglecting the *small_const*).

Context-dependent similarity mode - similarity in terms of some subset of components - is also possible. For example, if we are interested in similarity in terms of ***A*** and ***B***, similarity of ***X*** and ***Y*** is full.

3.3.3 Decoding or unfolding

Decoding or unfolding is finding the components provided with the representation of a composite item. In the fully-articulated local representation, information on the components is immediately available from the interpretation of activity of individual units'. In the disjunctive distributed representation, the similarity of a composite codevector to each of its possible component codevectors is calculated, and the component codevectors with the maximal values of overlap are selected. For example, **AvBvC** has the value of similarity of approximately 1/3 to each of **A**, **B**, and **C,** and much less similarity to other possible component codevectors.

3.4. Complex composite items: superposition catastrophe and binding

Complex composite or hierarchical items are the items whose parts are in turn compositional items.

3.4.1. The superposition catastrophe

If we try to construct representations of hierarchical items recursively, by superposition of their fully-articulated representations, we will encounter the problem well-known as "ghosts" or "superposition catastrophe" (Feldman 1989; von der Malsburg 1986; see also Rachkovskij & Kussul 2001). It is connected with the fact that superposition is not an adequate operation to compose representations of hierarchical items.

Under superposition, representations of the parts composing the whole are not bound together. The fully-articulated representation of the whole is indistinguishable from the representation of the set of their parts. Therefore, when several composite sub-items are combined into the new whole, their components are mixed together, the information on their combination in sub-items is not preserved, and it can not be determined to which whole a part belongs.

The simplest illustration could be as follows. Let there be components ***A,B,C*** and composites ***AvB, AvC, BvC*** (Figure 3.3). If the codevectors of any two of three composites (e.g., ***AvB*** and ***BvC***) are activated, the third composite will become active as well, though unbidden (here, "ghost" ***AvC***). In "superposition catastrophe" formulation, the same situation is described as follows: which two composite items are really present, if ***A,B,C*** are activated? These problems also manifest themselves as "spurious memories" (Hopfield, Feinstein, & Palmer 1983; Vedenov et. al. 1987) for superposition learning rules, e.g., Hebb's learning rule (Hebb 1949), in matrix-type distributed associative memory (i.e., spurious attractors in Willshaw or Hopfield networks).

3.4.2. Binding

In order not to confuse the component items of different composite items, and thus to overcome superposition catastrophe, representation of the whole should be different from superposition of its parts' representations.

In symbolic bracketed notation, representation of hierarchical items may be thought of as expanded in time or space and structured by brackets. The brackets show how the components are grouped or bound together. In *((AB)(AC)), A* is bound to *B, A* is bound to *C, AB* is bound to *AC,* where *A, B, C* are elementary items.

To transform symbolic bracketed notation of structured items into local or distributed connectionist representations, some analog of brackets is required that shows the boundaries of sub-items. The mechanism for introduction of such "brackets" or grouping together representations of items is called binding.

Let us consider "conjunctive" schemes for implementation of the binding operation.

3.5. Conjunctive binding and reduced representation of simple composite items in local representations

3.5.1. The binding scheme

In *local* representations, introduction of a new unit to represent a composite item is analogous to introduction of brackets to represent binding of components, or to introduction of a new symbol (to substitute) for a combination of symbols denoting bound parts inside the brackets. Such a local representation of composite items may be called reduced or conjunctive local representation, or local conjunctive binding of the components. For example, *A&B&C -> (ABC).*

The introduced unit representing the binding of component units has connections with the latter. These connections represent part-whole relations between the components and the composite (see also 3.7.1). Thus, introduction of local reduced representations of composite items leads to the growth of the units' pool and necessity to beat connections to the introduced units (Figure 3.4).

3.5.2. Decoding and similarity of bindings

In local reduced representations, decoding of bindings (or retrieval of component representations) is done by following of connections from the unit representing the composite to the units-components (Figure 3.4).

In *local* representations, each composite item represented in the local reduced form makes up a machine for calculation of its similarity value to any other composite item represented in the fully-articulated form - a dot-product machine. This value is available from the unit of reduced representation. The connection weights (equal to 1 for binary representations) encode its components, e.g., ***(ABC)=A&B&C*** in Figure 3.5. Components of the fully-articulated local representation of another composite item are activated in the input component pool, e.g., ***AvBvD***. Then the activity of the ***(ABC)*** unit provides the readings of the similarity value S between two simple composite representations, ***AvBvC*** and ***AvBvD*.** In our example, S=2/3.

However, content of similarity is absent at the unit of reduced representation. Obviously, the same value of similarity may be produced by very different composite items, e.g., *ABC* and *ABD, EBC, AFC,* etc. Also, if the component units are activated differently, the contribution of each unit is undefined - only the sum of contributions is available.

In order to find the similarity content of two simple composite items provided with their (non-identical) local reduced representations, both should be decoded through their fully-articulated representations, and then the elementary components should be compared (by elementwise product). To find their similarity value, one of the simple composite items should be decoded, and then the dot-product machine of the local reduced representation of the other composite item will output the value of similarity.

However, if the similarity value of a unit corresponding to reduced representation is not maximal (the items are not identical), then there is not enough info for retrieval of the components - it is not known, which subset of components provided the observed value of similarity.

3.6. Conjunctive binding in distributed representations

3.6.1. The binding schemes

Various schemes are used for conjunctive binding of distributed representations (for a review, see, e.g., Rachkovskij & Kussul 2001; Plate 1997). Binding by tensor product $X_{ijk} = A_iB_jC_k$ (Smolensky 1990) gives the growth of dimensionality of the binding X_{ijk}. In (L)RAAMs (Pollack 1990; Sperduti 1994),

dimensionality of binding codevector may be the same as the dimensionality of component codevectors, but it must be learned by training of a multilayer perceptron.

Some binding procedures for distributed representations have been proposed that do not require training and preserve dimensionality of components. They are convenient for construction representations of nested compositional items. Such binding operations include binding of real-valued codevectors by circular convolution proposed by Plate for the Holographic Reduced Representations (Plate 1991; 1995) or binding by elementwise multiplication proposed by Gayler (1998). For binary representations, binding by elementwise XOR is used in Binary Spatter Codes of Kanerva (1994; 1996).

In APNNs, the Context-Dependent Thinning procedure (CDT or thinning) is used for binding and allows preservation of density of bound codevectors (see section 6 and Rachkovskij & Kussul 2001). In distinction to the result of superposition, where the component codevectors are represented in full form (by all of their 1s), in bound representations they are represented in a reduced form (by a fraction of their 1s). The reduced representation of any component codevector depends on the other component codevectors from the composite. Therefore, the information to which group of components a particular component codevector belongs is preserved (Figure 3.6).

3.6.2. Decoding of bindings

In distributed representations, retrieval of the component codevectors provided with the reduced representation of their binding is done by "unbinding". In APNNs, it is done by finding the component codevectors most similar to the thinned codevector (in the same way as for the representation by superposition, section 3.3.3). For this purpose, one should have a memorized set of all the component codevectors.

3.6.3. Similarity of bound reduced representations

Unlike local reduced representations, where a binding unit does not carry immediate information on the content of its components, but carries only connections to them, in distributed reduced representations the codevector of binding carries immediate information on its component codevectors (e.g., Hinton 1990)..

In APNNs, a thinned composite codevector is of the same dimensionality as component codevectors and contains subsets of their 1s. Therefore, binding by thinning preserves the similarity of the reduced representation of the whole to full representations of its parts. Besides, the reduced representations of similar sets of items are similar to each other. For example, <***AB***> is similar to <***AC***>, since the reduced representation of ***A*** in both of them are similar. (Angular brackets denote distributed reduced representations produced, in particular, by the thinning procedure). Both for APNNs and for other distributed representation schemes, <***AB***> is similar to <***A′B′***> if ***A*** is similar to ***A′***, ***B*** is similar to ***B′***.

These properties of reduced distributed representations allow an easy estimation of similarity of composite items, in the same manner as for elementary items. The content similarity is estimated by the bitwise conjunction of the thinned composite codevectors, and the value of similarity is estimated as the fraction of 1s in this content codevector (as in section 3.2.1). This does not require finding of the fully-articulated representation, as for local reduced representations, but is done by the immediate use of the reduced composite codevectors. The overlap of reduced representations contains the information both about the value of similarity and the set of overlapping components (content). Also, the overlap can be decoded, because it contains representations of common components.

In local representations, each local unit provides a specialized machine for calculation of its similarity value to a fully-articulated representation of any input item (Figure 3.5). In distributed representations, a universal machine can be used for this purpose (Figure 3.7).

3.7. Hierarchical or complex composite items

3.7.1. Representation

The procedure of conjunctive binding allows superposition catastrophe to be avoided when representing complex composite items.

Local reduced representation of hierarchical composite items is done by introducing binding units of higher hierarchical level connected to the binding units of lower hierarchical levels. Such

"growing" local connectionist networks have been developed by a number of researchers, such as Quillian (1968), Amosov et. al. (1973), Gladun (1977), Grossberg (e.g., 1982); Carpenter & Grossberg (1987), etc.

In *local* connectionist networks, a composite item can be represented as a tree or Directed Acyclic Graph (e.g., Frasconi et. al. 1997), where items are represented by units (Figure 3.8). Each combination of components is represented by a separate unit, so that the groups of parts belonging to different wholes do not get confused.

For *distributed* encoding, reduced and bound representations of composite items can be also recursively bound with each other to form a more complex composite representation. However, it is rather difficult to depict such a representation, as Figure 3.6 demonstrates. This is probably one of the reasons why localist representations got more popularity.

3.7.2 Decoding of reduced hierarchical representations

Both local and distributed reduced representations of hierarchical items can be decoded (expanded, unfolded) through fully-articulated representations of their components. However, fully-articulated representations must be properly grouped with each other, as in the initial bracketed notation considered above (section 3.4.2).

Decoding a composite item representation is done by sequential decoding of the higher level bindings through the lower level component bindings, down to representations of elementary items. If there is a single pool of units encoding elementary items, the structure of a composite can be unfolded over time by sequential activation of elementary items' representations (e.g., Hummel & Holyoak 1997). Appropriate grouping of items is done by hierarchical organization of items' activation in time. Items of certain group are activated in adjacent time moments.

So-called dynamic binding - binding a group of elementary items by their synchronous activation - may be considered as a special case of binding by sequential activation, when a sequence of items is replaced by their synchronous activation (Milner 1974; von der Malsburg 1981; Shastri & Ajjanagadde 1993; Hummel & Holyoak 1997). For the example of Figure 3.3, the units (for local representations) or distributed patterns (for distributed representations) encoding *A* and *B* are activated at one moment. Representations for *A* and *C* are activated at the other moment. Binding of *AB* and *AC* is represented by sequential activation of just mentioned representations. Items adjacently activated in time are considered belonging to a single group of items. The interval between activation of representations serves as an analog of brackets. Probably, activation of special "bracket" nodes could also be used. In principle, *A* and *B*, *A* and *C* could all be activated sequentially, but with different intervals between them (Figure 3.8b). The sequence of activation patterns in time can be represented in space, if each distinct pattern is represented over its own copy of the units' pool.

3.7.3. Decoding of local representations

To decode local reduced representations of hierarchical items, all branches of their local representation trees should be sequentially followed, starting from the unit of the top level through the intermediate levels down to the units of the bottom level (Figure 3.8). Adjacent branches of each hierarchical level are traced sequentially, in adjacent moments. Thus the elementary items of each bound group are activated in adjacent time moments, and the activation of each non-elementary unit may be considered as representing the tags or grouping brackets of certain nesting level (section 3.4.2).

In such a way, a dynamic version of representation of composite items by their elementary items can be realized for local representations.

3.7.4. Decoding of distributed representations.

Reduced distributed representations can also be decoded through full distributed representations of elementary items. For this purpose, as in local representations, bound items of each hierarchical level should be decoded through their subitems of the lower level, down to the base-level. For this purpose, unbinding is used (section 3.6.2). For binding by the CDT procedure, decoding is done by finding the components of the lower level most similar to the binding. As in local decoding, activation of a component item of some level serves as an analog of brackets.

Thus, unbinding (e.g., by finding similarity) in distributed representations serves as an analog of connection following in local representations.

3.7.5. Hierarchical memory organization

In APNNs, a hierarchically organized memory is used. The codevectors of reduced representations of composite items of each level of part-whole hierarchy are stored in separate memory arrays. This allows us to distinguish the processes of finding most similar codevectors of the same complexity and the processes of finding part by whole and whole by part.

For example, ***A,B,C,D,*** etc. are stored in one array, **<*AB*>, <*CD*>,** <***ABCD***>**,** etc. are stored in another array, and **<<*AB*><*CD*>>, <*A*<*ABCD*>>,** etc. are stored in the third array. If we have a probe <AB> of the intermediate hierarchical level of part-whole hierarchy, we can find the most similar codevector of the same complexity by searching the intermediate-level memory array. To find its components, ***A*** and ***B***, we must search the lower-level memory array. And to find the super item **<<*AB*><*CD*>>**, we must search in the memory array of the third level. Otherwise, if all the codevectors were stored in the same memory array, we could not distinguish, whether we retrieved a part, a superset, or an item of the same complexity.

3.8. Similarity of complex composite representations

3.8.1. Similarity of fully-articulated representations of hierarchical items

The estimation of similarity between fully-articulated representations of composite items is a difficult problem. These issues are under investigation in the models of human analogical reasoning (see, e.g., Gentner 1983; Falkenhainer, Forbus, & Gentner 1989; Gentner & Markman 1995, 1997; Hummel & Holyoak 1997; Eliasmith & Thagard in press). The total similarity of structures (composite items) should take into account not only the similarity of elementary items ("semantic similarity"), but the similarity of their grouping ("structured similarity") as well.

Let us consider simple examples of composite items:

((ABC)(DEF)(GH)) (3.3)
((CE)(FGH)(ABD)) (3.4)
((ABCDE)(FGH)) (3.5)
((AB)(CD)(EF)(GH)) (3.6)

Unlike the structured propositions used in analogical reasoning, here we do not take into account the sequence of items. The sets of elementary items of these structures are identical (exact semantic similarity). However, the items are grouped differently. The similarity content and therefore the similarity value depends on grouping of elementary items and correspondences or mappings of those groups between two compared structures. Since it is unknown in advance what is the correspondence between subitems, it is generally required to obtain the values of similarity for all possible mappings of substructures and choose the mapping with the maximal similarity value (Figure 3.9). In order to avoid exhaustive search through all possible mappings, some constraints are introduced on possible correspondences which reflect peculiarities of human analogical mapping. Sometimes constraint satisfaction connectionist networks are used to find the best mapping and the corresponding similarity value (Holyoak & Thagard 1989; Falkenhainer, Forbus, & Gentner 1989; Eliasmith & Thagard in press).

As an example, let us consider the scheme for similarity estimation of two composite items similar to that used by Hummel & Holyoak (1997). In this scheme, the composite items are first represented as trees of local bindings. Then, in the process of mappings and estimation of similarity, branches of representation trees are used as the machines for finding similarities between items of various hierarchical levels.

Let us consider the composite items of Eq.3.3 and 3.4. Represent them as the local trees (Figure 3.10). Unfold the composite item of Eq. 3.3 in time, (section 3.7.3). *C* of *(CE)* sets the activation of the unit *(ABC)* to 1/3, and *E* sets the activation of the unit *(DEF)* to 1/3. The first subitem is finished, so we must preserve the unit with maximum activation. Since there are two units with the same activation, let us first preserve the activation level of the unit *(ABC)* and reset the unit *(DEF).* Then comes *(FGH). F* sets the activation level of *(DEF)* to 1/3, and *GH* sets the level of *(GH)* to 1. We preserve *(GH).* Then comes *(ABD)* that is automatically mapped to *(DEF)* and gives the activation level of 1/3.

Now all the units of the lowest level are reset, and thus we obtain the activation levels for the three units of the intermediate level for the considered mapping:

(CE) -> (ABC) = 1/3
(FGH) -> (GH) = 1
(ABD) -> (DEF) = 1/3.

It gives the activation level of 5/9 for the highest level unit (reduced representation of the whole structure) that provides the total value of similarity for the structures. However, it can be easily seen that the mapping

(CE) -> (DEF) = 1/3
(FGH) -> (GH) = 1
(ABD) -> (ABC) = 2/3

provides the value of similarity 6/9 > 5/9.

Thus, finding similarity in *local* or symbolic representations requires putting into correspondence subitems of composite items. If the trees of the items are not fully isomorphic, it generally requires consideration of various mapping versions and calculation of the total similarity value for each version. This is a computationally expensive process.

For *distributed* representations, if fully-articulated representations of both items are given, they are used for construction of codevectors of their reduced distributed representations. These representations are built independently from each other. Then their similarity content can be found as overlap of the codevectors, and their similarity value can be calculated as the fraction of overlapping 1s.

Reduced representations of subitems are incorporated in the reduced representation of the composite item. Since reduced representations of sub-items are formed from the reduced representations of their own component items, the components and their arrangements of the lower hierarchical level influence the code of complex composite items at the higher levels. Thus, the overlapping 1s of the resulting (high-level) reduced representations "automatically" find mapping of reduced representations of subitems, taking into account not only the similarity of their components, but the similarity of their part-whole arrangements as well. Mapping of subitems and calculation of their similarity value is not required for finding the total similarity of composite items.

In case explicit mapping of sub-items of two distributedly represented structures is still required (e.g., for analogical mapping), it can be done as well (see, e.g., Rachkovskij 2001).

3.8.2 Similarity of two items, represented in reduced form

For *local* reduced representations, two non-identical composite items (e.g., Eq. 3.3. and 3.4) are represented by two active units at some hierarchical level of the representation tree. It is impossible to estimate the value and content of their similarity immediately. One of the reduced representations must be decoded to the fully-articulated representation, and then similarity to the other representation should be found as in section 3.8.1, taking into account their mapping.

For *distributed* reduced representations, two composite items are represented by their respective codevectors. The content and the value of their similarity can be immediately estimated by their overlapping 1s and their fraction (dot-product). It takes into account not only similarity of the sets of elementary items in the composite item, but the similarity of their arrangements as well. That is, both semantic and structured similarity of the composite items is taken into account. So, decoding reduced representations, finding fully-articulated representations, and mapping the component items is not required for finding similarity.

3.9. Working memory and long-term memory

Working memory serves for construction, temporal storage, processing of a small number of item representations. Therefore, in previous sections we considered operations in working memory. Representations in working memory can be compared, bound, unbound. They can be transferred to long-term memory and stored there, or used for recall from long-term memory. Information in working memory is stored in active form - as activation of units or temporal connections between them.

For *local* encoding schemes, simultaneous representation of several composite items may require several pools of input units encoding elementary items, and also a pool of uncommitted units to represent bindings or reduced representations of various complexity. Representation of a binding is

realized by beating temporal connections from the units-parts to the unit-whole. In order to use the same working memory to represent new items and forget "old" ones, connections must be modified promptly, and so the meaning of non-elementary units can be modified as well.

Instead of several copies of units' pools, time sharing of single pool of units among different items can be used. Some mechanisms for duplication of units and connections of one pool to another, including the long-term memory pool, should also exist.

In *distributed* schemes, working memory is a number of encoding pools - neural fields of the same dimensionality - for temporal storage and processing of codevectors corresponding to items of various complexity. Processing can be superimposing, thinning, finding similarities, differences (common and different 1s), and other elementwise operations.

In distinction to working memory, long-term memory permanently stores an abundance of item representations - the whole world model. Usually it is assumed that the information in long-term memory is stored in passive or inactive form - as long-term connections between the units.

Information retrieval from long-term memory occurs by the activation of units through connections. The units of the long-term memory pool may be considered as one of the working memory pools (at least the pool of elementary items). The units encoding some currently observable item should be marked (tagged) somehow, e.g., by a high level of activation.

3.10. Long-term memory: local representations

3.10.1. Storage

Storage to long-term memory is transferring there representations from working memory. For local representations, let us assume that long-term memory is organized in the same way as reduced representations in working memory - that is, as trees of bindings encoded by units and their interconnections.

When information is transferred from working memory to long-term memory, units available in long-term memory should be re-used, at least for identical items. Otherwise, different units for the same item will appear in long-term memory.

Such a storage operation may be as follows. Long-term memory and working memory must have a common pool of units representing elementary items. The reduced representation of the composite item to be stored is decoded through the fully-articulated representation unfolded in time (section 3.7.3). Then, each group of elementary items is input to long term memory (as in finding similarity of two items, section 3.8).

Usually, each group of elementary items activates a number of units from the long-term memory pool, corresponding to the reduced representations of most similar composite items. If there exists a unit with a maximum or near maximum activation, it means that it gives a good fit to the input combination of items, and it can be used as the representation of the input composite sub-item. If there is no such a "close match" unit, a new unit should be allocated or introduced to represent the input items. The similarity threshold for introduction of a new unit is determined by the "vigilance" parameter (e.g., Carpenter & Grossberg 1987). If a unit with a non-maximal activation is used, the item is represented inexactly. For example, if the *(ABCD)* unit exists in long-term memory, it can represent *ABCE* or *ABCF* at the vigilance threshold less than 3/4. Existing or introduced units are used for constructing representations of more complex items.

3.10.2. Retrieval

For local representations, finding the unit representing a composite item in long-term memory that matches the input "probe" composite item generally requires separate comparison of the probe with the representation tree of each item from long-term memory. Otherwise, one can be trapped in a "deadlock" by activating the units that belong to the trees of different higher-level items, or that are not branches of the trees of higher levels at all.

Therefore, it is necessary to copy from long-term to working memory the representation trees of composite items that are the candidates for comparison, and find their similarity to the probe one-by-one (sequentially), using the procedure of section 3.8. Since the number of items in long-term memory is tremendous (may be many millions), this process could be prohibitively long, and some ways to limit the number of candidates must be used. For example, the candidates of the same complexity level with the probe could be selected, or the candidates that have a number of elementary items common with

probe, however without taking into account their groupings (e.g., Forbus, Gentner, & Law 1995; Thagard, Holyoak, Nelson, & Gochfeld 1990; Gray, Halford, Wilson & Phillips 1997).

Though such a two-stage scheme of finding the best match (selection of candidates, and then their sequential extraction to working memory and finding detailed similarity to the probe taking into account the structure) provides computational savings, it is still too expensive. Also, the number of candidates to be selected is not defined, and there is a risk to miss potentially best matching candidate.

3.10.3. Specific and generalized items and classification hierarchy.

If the vigilance threshold (section 3.10.1) is high, each new composite item will be represented by a new unit. If the vigilance threshold is not maximal, several existing units can get approximately the same activation from the input. Then the problem arises, which of the activated units should be used for further learning (construction of representations of more complex composite items) at higher levels.

If all activated units are chosen for further learning, then the higher-level units representing each of the combinations including each of the activated units must be created. In Figure 3.11, *ABCG* activates units *X=(ABCD), Y=(ABCE), Z=(ABCF),* and some other group of elementary items activate *U, V, W*. Then the higher level units *(XU), (XV), (XW), (YU), (YV), (YW), (ZU), (ZV), (ZW)* must be introduced or activated, and used for further learning at the higher levels. The combinations including all those units as parts of more complex items should be constructed and memorized as well, and so on. The number of such combinations is large and grows combinatorially with the number of composition levels (Figure 3.12), so their memorizing requires impractical number of units and time. If a single unit is selected for further learning from several activated units, the problem is which one to select, and that other activated units will not be taken into account.

Generalization looks like a better solution. Under generalization in local representations, several activated units representing similar composite items sharing common component items are considered belonging to the same class. The class item is generalization of specific items with common components and is represented by a newly introduced unit - that of concept or prototype. This class or concept unit is used instead of the group of specific item units in construction of more complex composite items. The concept unit is activated when at least one of its subunits is substantially activated, or, if its characteristic component items are activated. Thus, connections representing the "is-a" or "class-instance" relationship can be considered of OR type, to distinguish them from the AND type connections in the "part-of" or "part-whole" relationship. For our example, *ABCD, ABCE, ABCF* can be represented by a single concept unit *(ABC)* which is used for learning at higher composite levels (Figure 3.10).

3.10.4. Problems with generalization

Generalization with local representations gives rise to a number of problems that must be solved in order to make generalization useful.

(a). If a class unit is introduced or activated instead of some specific-item unit and is used in construction of composite items, the information concerning peculiarities of the specific item at the input is lost. That is, classification merges the specific items that may be important by themselves.

(b). How the items must be grouped into classes? A class is defined by its concept or prototype that has some combination of components common to its instances. How such a concept could be extracted?

To extract a concept from specific items, the following scheme may be used. When a unit is introduced to represent an item, let us introduce another unit with lower vigilance threshold to represent its possible class (concept). Let us increase its connections to the component units each time this unit is activated. Then, some connections corresponding to frequently met components will become stronger, corresponding to the concept.

For example, if an additional unit is introduced for the *ABCD* item (Figure 3.11), then input of *ABCE, ABCF* will activate it and enhance connections to the "potential concept" unit. Then, connections from the units *A,B,C,* to the concept unit will become stronger than connections from *D,E,F*. Therefore, this unit will be activated much more easily for the composite items with the components *A,B,C,* and it may be considered as the concept or prototype unit of the class *ABC*.

However, for such a scheme the number of "potential concept" units may be much more than the number of items, and connections should be modified from every input composite item to many

"potential concept" units. The reason is that though the number of classes seems to be less then the number of items, we do not know in advance which of "potential concept" units will actually extract a regular combination of component items that will form a concept.

(c). The same input composite item may belong to many classes. For example, *BCDG, BCDH, BCDI* will form the *(BCD)* concept. Then *ABCD* will activate both *(ABC)* and *(BCD)*, and, probably, *(ACD), (AB), (AC), (AD), (BC), (BD), (CD), (A), (B), (C), (D),* if such prototypes exist.
Therefore, the concept units for each class of the input item should be modified, and several concept nodes must take part in construction of higher level items, making the situation similar to the initial one, when we had several activated instance units for a single combination of items at the input.

(d). How the hierarchy of classes or concepts should be built? The items *ABCD, ABCE, ABCF* have the concept *(ABC),* whose unit should be introduced. And *BCDG, BCDH, BCDI* have the concept *(BCD).* However, these two classes also have a common class of a higher generalization level - the class with the concept *(BC).*

So, in parallel with feeding the input items to the units representing *(ABC)* and *(BCD),* they must be also fed to the *(BC)* unit, as well as to many other units. The vigilance threshold for more general classes must be lowered. The problem is how to connect less general class units (with more component items corresponding to attributes) to more general class units in order to form a hierarchy, and how to modify these superclass-subclass-instance connections when new classes are introduced. Learned connections from the instance units must be transferred to the concept units. Various operations can be proposed to modify the structure of connections between and outside the instance and concept units of various generality (e.g., Gladun 1977; 1994). However, those operations are inflexible, "rule-based", and require complex chains of set theoretic operations to be implemented.

Thus it seems that building the hierarchy of generalization or classification requires further growth of the number of units ("candidate" and "real" concepts). A number of concept units from different classes, that are activated for each input composite item, may be considered as a kind of distributed representations. In case of activation of several concept units, a single one with the maximal activation must be selected depending on the context, in the same manner as for an input composite item.

(e). How to vary the level of generalization, changing from specific exemplars to classes and back, and how to use the context to choose a certain instance belonging to a class?

Thus, it does not seem that the problem of unsupervised formation of generalization hierarchies has an easy solution for local representations.

3.11. Long-term memory for distributed representations.

For long-term storage and retrieval of distributed representations, auto-associative memory is usually used which can be distributed or non-distributed. In *non-distributed* memory, each codevector representing some item is stored in a separate memory word. In neural networks, it can be represented in the weights of connections from the active units corresponding to 1s in the codevector to the unit locally representing the item. Such an organization of long-term memory for distributed representations is similar to a set of similarity finding machines for local representations (section 3.5). A possible implementation of this type of memory may be that of Baum, Moody, & Wilczek (1988) or Carpenter & Grossberg (1987). To retrieve a codevector from such memory, the unit with maximum activation (the closest one to the input codevector) projects its activation to another pool through the weights that encode the item stored by the unit. The number of connections needed is $L*N$, where L is the number of items stored.

Distributed auto-associative memory (e.g., of matrix type) can be used with distributed representations. Its neural network representation could be feedback-type neural network, such as Hopfield (1982) or Willshaw (1981) memory. Each connection is a matrix element. In order to store a codevector, the weights between the active units corresponding to the 1s of the codevector are increased (Hebb's learning rule, see also section 5). Stored codevectors become attractors. To retrieve the "closest match" codevector, the probe codevector is input to the network which evolves to the nearest attractor.

In distributed memory of the matrix type of N units and NxN connections, $L>>N$ codevectors can be stored and retrieved successfully for the case of random sparse codevectors and the Hebbian

learning rule. Besides, the retrieval time does not depend on the number of stored codevectors. Its merits also include natural means to build the generalization hierarchy without the teacher (section 5), allowing the problems immanent for local representations (section 3.10.4) to be solved. Distributed auto-associative long-term memory in APNNs, in the same manner as working memory, is made up from separate memory arrays (section 3.7.5) corresponding to various levels of compositional complexity and various modalities (section 4).

3.11.1. Storage of representations of composite items

Storage of composite items to long-term memory requires storage of the codevectors of their components to the memory arrays of corresponding levels, as well as storage of the reduced representation of the whole item to the memory array corresponding to its hierarchical level.

One of the most important peculiarities of distributed memory is that during storage of a codevector, it is not needed to check if this or similar codevector is already present in memory (as required for non-distributed memory). In distributed memory, the same codevector modifies the same connections in memory. For example, for a superpositional learning rule and gradual connections (e.g., original version of Hebb's learning rule), a repetitive storage of the same item will enhance its memory trace (connections between its active units). Therefore, frequently occurring items produce stronger memory traces.

In non-distributed memory, where each item is stored separately (both for local and for distributed representations), each storage act requires making non-trivial decisions on which memory word(s) should be modified by the input item.

3.11.2 Retrieval - finding the closest match

For distributed representations, finding the most similar composite item in long-term memory (either distributed or local) does not require extraction of all items' representations from long-term memory and performing a complicated process of finding probe's similarity to each of them, as in local representations (section 3.10.2).

Simply, a reduced representation codevector of the probe composite item is formed in working memory. Then, it is used immediately for associative recall of the closest match (most similar) codevector from long-term memory array of its level of compositional hierarchy. The similarity of reduced composite representations by the maximal overlap of codevectors takes into account both the similarity of elementary items' sets and their arrangements. Moreover, the similarity of elementary items is not all-or-none, but gradual.

Also, a similar set of component codevectors is able to activate similar, but non-identical representations of higher levels, thus providing generalization without introduction of class representations (as needed for localist representations, section 3.10.3).

For distributed representations, there are two ways of forming reduced representation of a composite item-probe and finding the closest match. For the first mechanism, reduced representation of the whole composite item-probe is constructed in the working memory, without interaction with long-term memory. Then the closest match at the proper level of long-term memory hierarchy is determined. Such a mechanism may be called "postponed similarity finding". It is absent for local representations, because they require mapping between subitems of the probe item and those extracted from long-term memory at each hierarchical level, and finding their best combination that provides the maximum similarity to the probe.

Another way consists in finding most similar subitems at each level. It may be useful if we need finding not only the most similar composite item as a whole, but its most similar subitems as well. Also, we may use the most similar items which we found at each level, or the initial ones, to construct representation of the higher level.

An analogous scheme may be used for local representations. However, here we need not extract each of the long-term memory items to working memory in order to find the similarity with each of them separately. We can't get to deadlock because we move upward not by fixed connections in long-term memory, but by similarity. On the other hand, we may deflect from the input item or from the closest match if we find in long-term memory the most similar item to each of the input subitems. For example, let us have *(ABCD), (WXYZ)* at the intermediate level of long-term memory, and *((ABCD)(WXYZ)), ((ABCE)(VXYZ))* at the highest level. *((ABCE)(VXYZ))* is the probe. If we find the items of intermediate level closest to each of the input subitems, we get *(ABCE)->(ABCD), (VXYZ)->(WXYZ)*. Then we use them as parts for constructing of the top level item, and get *((ABCD)(WXYZ))*

as the closest match. However, using the scheme of "postponed similarity finding", we get *((ABCE)(VXYZ))* as the closest match at the top level, which is identical to the probe.

3.11.3. Decoding

Reduced representation of a composite item can be decoded through reduced representations from subitems of the lower-level arrays of long-term memory. Reduced representations of specific composite items may be decoded through their specific components, not through the classes, as needed for local representations.

If a composite item is activated from higher levels, then decoding allows finding the most close components in lower-level arrays of long-term memory. If an item is activated from the lower levels (a probe of the same complexity or a component), and its subitems are present in the memory arrays of the lower levels, decoding allows a detailed investigation of its composition, finding similarities and differences of its components to those of the probe, predicting unobservable components.

Unlike local representations, where decoding is simply following the pointers or connections, in distributed representations decoding is done based on similarity. From one hand, this process may be more complicated and ambiguous, because it potentially requires selection of a component from a number of similar ones. From the other hand, it allows decoding of a composite item that is not present in long-term memory (e.g., that from working memory). It also allows retrieval of various versions of the same subitem, taking into account the context of the composite item during decoding.

It is also possible to retrieve and decode subitems with various degree of generality. For example, given *((ABCD)(EFGH)), (ABCD), (ABC), (AB)* may be retrieved. Also, *((ABCD)(EFGH))* may look as *((AB)(EF))* at certain level of generality.

3.11.4. Generalization and classification hierarchy

Distributed memory can "extract" some regularities in representations of items - frequently occurring combinations of 1s in their codevectors. These regularities may correspond to some objects, situations, etc., or to their classes represented by concepts or prototypes.

Since similar items have overlapping representations, in distributed memory with the Hebbian learning rule the memory traces corresponding to frequently occurring parts of representations (i.e., connections between the 1s frequently occurring together) become stronger than those of rare combinations. For example, if *ABCD, ABCE, ABCF* are input, the weights between the units encoding *A, B, C* and between the units of item pairs *A* and *B*, *A* and *C*, *B* and *C* become larger than the weights between the units encoding *D, E, F* and the weights from those units to the units of *A, B, C*. Therefore, representation of *ABC* can be easily activated.

The items *BCDG, BCDH, BCDI* will generate concept *BCD*. However, the connectivity of *BC* will be stronger than the traces of either *ABC* or *BCD*, and *BC* can be activated even easier. Thus, after learning of items *ABCD, ABCE, ABCF, BCDG, BCDH, BCDI*, we "automatically" obtain a hierarchy of similar items corresponding to classification hierarchy. *BC* is the most abstract concept, *ABC* and *BCD* are less abstract concepts, *ABCD, ABCE, ABCF, BCDG, BCDH, BCDI* are instances. Such classification or generalization hierarchies can be formed for composite items of various complexity. The same item may belong to various hierarchies. For example, *ABCD* may belong to classes with the concepts *ABC, ABD, ACD, BCD, AB, AC, AD, BC, BD, CD*, etc.

This ability for generalization appears in distributed memory because all items of the same complexity or memory array are stored in the same memory matrix (not in separated memory words, as for local memory). It provides a natural basis for building a classification hierarchy.

3.12 Specific items and context

In *local* representations, a specific item (a class instance)in reduced form is represented as a unit. If there is a concept unit of its class, then a specific item may be represented as the unit connected to the concept unit, and to the units of the component items that supplement the component items of the concept (Figure 3.13. This scheme is different from that of Figure 3.11, where instance items have all the components). If we decode a specific item, e.g., *(ABCFG)* or *(ABCFG)*, we must go to its class item *(ABC)* and decode it, as well as decode supplement components, *F, G* or *D, E*.

However, if there are many classes to which an instance belong, e.g., *(ABC), (AFG)*, etc., there should be as many units for the single instance item, each having a different set of supplement

components for its class, e.g., *F, G* or *B, C*. In the process of decoding, the context must choose the instance of a class, e.g., by activating some of its component units. From *(ABC)* we go to *(ABCDE)*, if *D* and *E* are activated below (Figure 3.13). If the bottom-up context (*D, E*) which specifies the instances is absent, and we used classes instead of instances to form composite items, we can't find the instance which produced the composite item.

For *distributed* representations, we can use components-instances to form a composite item, and still be able to activate the same composite item by similar, but non-identical instances.

For distributed representations, the representation of an item can be modified by context from top, bottom, or the same level. For example, an item may be represented only by some of its components, depending on the context. These components will be retrieved during decoding. These components also influence similarity. For example, in the context without *D* and *E*, *(ABCDE)* may be represented by *A,B,C.*

In distributed representations, we may change (include or exclude) not only the whole representation of a component, but its particular version (subset of representation) as well. For example, we may not only include or exclude the component *A*, but use very many of its versions as well (Figure 3.14).

As will be discussed in section 5, in the process of retrieval from long-term distributed auto-associative memory, the concept items or instance items *in particular version* can be activated depending on the input activation, the connectivity pattern (strength of traces) in long-term memory, the level of details, or fatigue of neurons and connections. In the process of decoding, specific realization of the item-whole may activate specific (context-dependent) version of the item-part (Figure 3.14).

In local representations, each stored representation is rather rigid and discreet. If a local representation is formed, it can be activated or not, thus corresponding to the set of its components. The units influence only the directly connected units. Each variant of item representation in particular context will require a separate unit (Figure 3.15). So, the context will also require the growth of the number of units. Also, the context is rigid - only very different contexts are represented by different units.

3.13 Summary: local vs distributed representations

Thus, the basic distinctions between local and distributed representations consist in:

- the way to represent similarity of items;
- the information capacity of representations;
- the storage capacity and other properties of suitable long-term memory scheme;
- the schemes for construction, similarity estimation, and decoding representations of composite items (the part-whole hierarchy);
- the methods of concept construction and varying the level of details in representations of (composite) items (the generalization or classification hierarchy).

These distinctions might be not so important for artificial domains, where processing a small number of simple composite items (with single composition level) suffice. Moreover, local representations may be preferable for building models of such simple domains, because they are more obvious and require less memory (e.g., Page 2001). However, examination of local representations' applicability for building models scalable to the size and diversity of the real world reveals their limitations.

Structured distributed representations look more promising for implementation of adequate world model since they make it possible:

- to reflect diversity of the world in diversity of distributed representations;
- to combine representations of components and introduce new representations without increasing their dimensionality;
- to make up reduced and bound hierarchical representations of composite items from reduced representations of their components, preserving immediate information on the components;
- to estimate the value and content of similarity of such complex composite representations immediately, without retrieval of their components, but taking into account their structure;
- to make and find associations between the wholes and their parts based on partial, context-dependent similarities of their representations (e.g., predicting unobservable components and superitems);
- to accumulate experience by storing representations in distributed long-term memory which has high storage capacity;

- to retrieve from distributed memory representations of composite items of various complexity which are context-dependent similar to the input probe;
- to extract the regularities of stored representations which may correspond to items or concepts and provide a natural basis for building generalization or classification hierarchy;
- to process concepts of various level of generalization or abstraction in the same manner as instances;
- to take context into account flexibly and without growth of the unit pool's dimensionality.

The remaining part of the article provides a more detailed description of the APNN architecture.

4. The skeleton structure of APNNs

In APNNs, the world model is constructed with binary sparse distributed representations. The APNN architecture is multi-module, multi-level, and multi-modal. Each module stores and processes model items of a certain level of compositional hierarchy and of a certain modality. Representations of composite items of a module are constructed from representations of component items of other modules that belong to previous levels of compositional hierarchies of single or several modalities. The modules corresponding to the bottom levels of compositional hierarchies contain distributed representations of elementary (base-level or indivisible) items.

Each module consists of several *neural fields* - pools of units or neurons (Figure 4.1). All neural fields have the same number N of binary neurons. Items are represented as distributed patterns of activity over the neural fields. These patterns or codevectors are stochastic, binary, and sparse (with a small fraction M/N of 1s, where M is the number of 1s). To get an impression on the values of M and N, M>1000, N>100,000 (see also section 5).

The fields are linked by bundles of N non-modifiable projective connections. Each projective connection from the bundle has a unity weight and connects its unique pair of neurons from two fields connected by the bundle. Each connection is assumed to be bi-directional and can be implemented by two unidirectional connections. Each connection can be in two states - conductive or broken. The state of all connections in the bundle is the same and is switched by a special control system. Apparently, connection break can be implemented with intermediate neurons, and a neural network implementation of the control system is possible.

Projective connections can be *direct* (connecting the neurons with identical numbers) or *permutive* (connecting the neurons with different numbers). Depending of the type of synapse, a connection can activate or inhibit a neuron. The bundle of direct projective connections with excitatory synapses transfer the pattern of activity from one field into another.

The main neural field of each module is an associative neural field. It is a distributed auto-associative memory of matrix-type that can be implemented by a full-connected feedback-type neural network. Unlike projective connections between the neural fields, associative connections within an associative field are modifiable. The associative field of each module not only performs the functions of long-term memory for its items (associative storage and retrieval), but it also performs the function of building generalization or classification hierarchies for those items (see also sections (see also sections 5.3, 2.1.5, 2.2.2, 3.11.4).

In addition to the associative field, there are a number of buffer fields in each module (Figure 4.2). A buffer neural field is able to receive and temporarily store patterns of activity from other fields. Thus the buffer fields perform the functions of working memory. Besides, the buffer fields connected with each other by various bundles of projective connections and having various threshold values can perform various operations over the distributed representations activated in them. The examples of such operations are superposition (bitwise disjunction), estimation of similarity and difference (bitwise conjunction and set theoretic differences). The most important operation of binding and normalization by Context-Dependent Thinning is also implemented in buffer fields (see section 6). A specific structure of buffer fields and their connections is used for a particular application or algorithm.

Various modules are connected with each other by bundles of projective connections between their fields (Figure 4.3). For simplicity, it is assumed that the fields of all modules contain equal number of neurons. It is possible that some fields of some modules consist of several concatenated fields of lower level modules, however, the possibilities of multiple use of concatenation are limited because of the field size growth.

Reduced representations of items of the same compositional complexity and modality, though maybe of different generality, are constructed, stored, and processed in the same module, whereas representations of different complexity or modality are in different modules. The reduced representation of a composite item-whole consists of reduced representations of its component items-parts - the structural parts with their relationships and/or the global attributes of the whole. Reduced representations of components depend on the other components of the whole, therefore representations of the components are bound in the reduced representation of the whole.

Representations of component items appear in the lower-level modules of the same or different modalities from sensors, or are retrieved from memory, or are constructed from the representations of their own components of still lower levels. They are transferred to the higher-level module, where the representation of the composite item must be constructed. In the buffer fields of that module, representations of the component items thin each other and make up the reduced

representation of the composite item, where the component representations are reduced and bound to each other (see also section 6).

Such a reduced representation of the composite item can be stored in the associative field of the module, or it can used as a probe to retrieve representations of similar items (instances or concepts), or it can be used for comparison with representations of other items in the buffer fields comprising working memory of that module. It can also be transferred to higher-level modules, where it can be used as a component item for construction of other composite items or as a probe to retrieve such super-items from memory. It can also be transferred to lower-level modules for decoding - retrieval of the fully-articulated representations of the components from their reduced representations in the composite item. Distinguishing higher-level and lower-level modules with respect to the current one allows us to control the direction of traversing part-whole hierarchy: from part to whole, from whole to part, the same composition level.

Representations of items from the compositional hierarchy of the same modality are stored and processed in "vertical chain" of modules. For example, representations of letters may be stored in one module, combinations of letters (syllables) may be stored in the higher-level module, and words may be stored in still higher-level module, etc. In visual recognition, representations of the roof, the wall, the window may be formed in one module, and representation of the whole house may be formed in the higher-level module.

Besides, representation of a composite item may be formed from the components of different modalities that may be transferred in parallel from several modules corresponding to those modalities. For example, visual model of an object is made up from attributes of shape, color, texture, size, position, movement, etc., represented in separate modules. Multi-modal sensory representation of an object is formed from its models of the modules corresponding to visual, acoustical, tactile, olfactory, taste modalities (or some of them). Named representation of an object is formed from representations of its name and sensory multi-modal representation. Implementations of the APNN architectures for different problems must comprise combinations of modules interconnected in both ways (serially and in parallel), see Figure 4.3.

Let us consider the structure and function of the APNN architecture in more details, beginning with the associative field.

5. The associative field: structure and functions

The associative field that realizes the functions of long-term memory in APNNs is based on Hebb's ideas about cell assemblies.

5.1. The paradigm of cell assemblies

In the late 1940s, Donald Hebb proposed a theory of brain functioning (Hebb 1949) that influenced the work of most researchers in the fields of artificial intelligence, cognitive psychology, modeling of neural structures, and neurophysiology. According to Hebb's paradigm, nerve cells of the brain are densely interconnected by means of feedforward and feedback excitatory connections, forming a neural network. Each neuron determines its membrane potential as the sum of active neurons' outputs weighted by connection efficacies. The neuron becomes active if this potential exceeds the threshold value. During network functioning, connection weights between simultaneously active neurons are enhanced. This results in consolidation of neurons into cell assemblies - groups of nerve cells most often active together - and leads at the same time to mutual segregation of assemblies. When a sufficient part of a cell assembly is activated, the assembly becomes active as a whole. At any moment, there is always only one active assembly. The assemblies may overlap, so that a single neuron may belong to many assemblies.

Formation of cell assemblies may be considered as a learning process, and the formed cell assemblies may be regarded as memorized internal representations of items or models that were encoded by the distributed patterns of active neurons. The process of assembly activation by its part may be interpreted as the process of pattern completion or associative retrieval of stored information when provided with its portion or a distorted version referred to as a memory key or a probe. Moreover, assembly neural networks have some degree of neurobiological plausibility, being parallel, distributed, noise and fault tolerant, etc.

Hebb's paradigm of cell assemblies - interpretation of various mental phenomena in terms of cell assemblies - has turned out to be one of the most profound and generative approaches to brain modeling and has been elaborated by Milner (1957, 1996), Willshaw (1981), Marr (1969), Kussul (1980, 1992), Goltsev (1976, 1996?), Braitenberg (1978), Palm (1982), Hopfield (1982), Kanerva (1988), Lansner & Ekeberg (1985), Amit (1989, 1995), Pulvermuller (1999), and very many other scientists.

5.1.1. Activity control in assembly neural networks

One of the problems in neural networks with exclusively excitatory connections is to maintain appropriate stable level of network activity, i.e., the fraction of simultaneously active neurons. Internal activity regulation by means of modifiable inhibitory connections was proposed by Milner (1957). However, development of this idea in Hopfield networks (Hopfield 1982) has resulted in low storage capacity for dense codevectors.

The networks with solely excitatory connections and neurobiologically relevant low activity level proved to be more promising (e.g., Willshaw, Buneman, & Longuet-Higgins 1969; Palm 1980; Frolov & Muraviev 1987;1988;. Amit 1989; Amari 1989; Tsodyks 1989; Nadal & Toulouse 1991; Frolov, Husek, Muraviev 1997). In such networks, the problem of activity control can be solved by an external system that controls activity level by generating an appropriate uniform influence on all neurons. The idea of such an activity control system was suggested by Amosov (1967), its possible neural network implementation was also proposed (Amosov & Kussul 1969) and tested in a hardware model (Goltsev 1976). The mechanism of uniform threshold control elaborated by Braitenberg (1978) turned out to be a more straightforward and convenient solution to this problem.

5.1.2. The internal structure of cell assemblies

The assemblies formed in the network with the Hebbian learning rule may have a very complicated structure, reflecting the history of learning. The connectivity between the neurons frequently activated together will be much stronger than its mean value. On the other hand, rare combinations of active neurons form parts of assemblies with a weaker connectivity. Hebb and Milner introduced the notions of "cores" and "fringes" (Hebb 1949, pp. 126-134; Milner 1957) to characterize qualitatively a complex internal structure of assemblies.

The notions of core (kernel, nucleus) and fringe (halo) of assembly that distinguish the parts of assemblies with a different connectivity have called attention to the function of assemblies distinct from the functions of auto- or hetero-associative memory. This function consists in accumulation of statistics on the frequencies of joint occurrence of attribute combinations encoded in the representations that form the assemblies. Common combinations of attributes are accumulated in the cores of assemblies which correspond to classes, whereas less frequent combinations are formed in the assembly fringes, corresponding to class instances. This function of assemblies provides the basis for emergence of classification (type-token) hierarchy (see also sections 5.3, 3.11.4, 2.1.5, 2.2.2).

5.2. The associative field

Let us consider an assembly neural network that is used as an associative field in the APNNs.

5.2.1. Auto-associative memory with binary neurons and connections

An associative field in its basic version is a fully interconnected network of binary neurons. Neuron output y is defined as

$$y[j] = \begin{cases} 1, \text{ if } I[j] >= t \\ 0, \text{ if } I[j] < t, \end{cases} \quad (5.1)$$

where t stands for the neuron threshold. An input sum $I[j]$ is calculated from

$$I[j] = SUM[i=1...n]\ y[i]W[ij], \quad (5.2)$$

where $W[ij]$ is the binary weight (0 or 1) of connection from the neuron i to the neuron j. The neural assemblies representing stored items are formed in the associative field in the process of learning using the binary version of Hebb's rule:

$$W[ij]' = W[ij] \vee y[i]y[j], \quad (5.3)$$

where $y[i]$ and $y[j]$ are the states of the i-th and the j-th neurons when the codevector $\boldsymbol{y}$ to be memorized is presented (i.e., the values of the corresponding bits of $\boldsymbol{y}$), $W[ij]$ and $W[ij]'$ are the connection weights between the i-th and the j-th neurons before and after training, respectively, v stands for disjunction.

At each step of iterative retrieval, the input sums of all neurons are calculated by equation 5.2, and the threshold t in equation 5.1 is chosen so that the number of active neurons in the network is approximately equal to some preset value. In the basic version, it is equal to the number of 1s in stored codevectors.

Such an associative field (equations 5.1 -5.3) is an auto-associative version of Willshaw memory. Behaviour of such a network is well-known. A probe codevector that is a partial or noisy version of one of the stored codevectors is input to the network. After few steps, the network comes to a stable state. Up to a certain number of stored codevectors, this state corresponds to the stored codevector closest to the probe.

5.2.2. The storage capacity and the size of an associative field

It is known that the number of random binary codevectors possible to store and retrieve in a distributed associative memory with the binary Hebbian learning rule grows under decreasing of the codevector density. The maximal number of codevectors is achieved for large networks and very sparse (M=log$N$) codevecors (Willshaw, Buneman, & Longuet-Higgins 1969; Palm 1980; Lansner & Ekeberg 1985; Amari 1989; etc.).

Even for not very large networks and for not very sparse codevectors (M=sqrt$N$), the number of codevectors can exceed N. For example, our experiments showed that the associative field of 4096 neurons can store and retrieve up to 7000 assemblies each consisting of 64 neurons (Rachkovskij 1990a, 1990b; Baidyk, Kussul, & Rachkovskij 1990). These numbers vary depending on the signal-to-noise ratio in the probe and on the probability of retrieval.

To build a large-scale model of the world, we need codevectors with a large number M of 1s. It is necessary in order to maintain a high information content of codevectors and statistical stability of

M in codevectors and their subsets. For example, *M*>1000 is required to provide the standard deviation of *M* less than 3%.

In order to ensure a reasonable sparseness of a codevector, the field size *N*=1,000,000 and the corresponding number *N*N* of associative connections required for a full-connected network are very large. Besides, the number *L* of connections per neuron equal to 1,000,000 is unreal from the neurophysiological point of view. Therefore, development of non full-connected ("diluted") networks is promising, e.g., with 10,000 connections per neuron. Thus, the dimensionality of an associative field may be of the order of *N*=100,000 for *M*<1000 or *N*=100,000,000 for *M*<1,000,000.

A number of experiments have been carried out to test the associative field with a diluted weight matrix (Kussul & Baidyk 1993b). Investigation of a rather moderate-sized networks (20,000,000 connections) has shown that storage capacity of diluted and full-connected networks is approximately the same when more dense codevectors (*M*>sqrt*N*, *M=0.01N*) are used in the diluted network.

5.2.3. Graduality of connections and the stochastic learning rule

The learning rule for binary connections (equation 5.3) does not form assemblies with a complex internal structure that are needed for emergence of generalization (type-token) hierarchy. The connectivity of an assembly becomes full after a single learning act and does not change thereafter. Gradual connections are needed to get a varied connectivity of assemblies:

$W[ij]' = W[ij] + y[i]y[j]$. (5.4)

In order to preserve the capability of forming assemblies with a non-uniform connectivity in the associative field with binary connections and binary distributed representations, Kussul proposed a stochastic version of the Hebbian learning rule (Kussul & Fedoseyeva 1987; Kussul 1992):

$W[ij]' = W[ij] \vee (y[i] \& y[j] \& x[ij]),\ mean(x[ij]) = r$, (5.5)

where *W*[ij]' and *W*[ij] stand for binary connection weight from the i-th to the j-th neuron before and after learning respectively, *y*[i] and *y*[j] are binary outputs of those neurons, v stands for disjunction, & stands for conjunction, *x*[ij] is a binary stochastic variable equal to 1 with the probability *r*.

After a number of learning acts, each connection takes certain value of 0 or 1, but the resulting assemblies possess complex structure very much alike the structure of assemblies in a neural network with gradual connections. The role of gradual connections is played here by the expectation *mean*(*W*[ij]) of binary random variables *W*[ij]. Probability of a connection between two neurons versus the number of their coactivations is given in Figure 5.1.

Since an item is represented by *M* active neurons, after a single act of learning with the reinforcement probability *r*, a neural assembly forms due to changes in O(*r*M*M*) of connections. After storage of a number of correlated codevectors, the neurons that have *M*'>*k1*M* (*k1*<=1)connections with the neurons of the same assembly may be attributed to the core part of the assembly, whereas the other neurons may be considered as the fringes (Figure 5.2).

The negative reinforcement -1<*r*<=0 leads to unlearning:

$W[ij]' = W[ij] \wedge \sim(y[i] \wedge y[j] \wedge x[ij]),\ mean(x[ij]) = |r|$. (5.6)

where ~ stands for negation. Under unlearning, the connectivity of an assembly already formed in the network weakens.

Stochastic versions of other learning rules, such as a *correlation* rule where connection weights between active and inactive neurons decrease (e.g., Pulvermuller 1999), are straightforward.

5.3 Correlated codevectors, prototypes, concepts, and generalization hierarchy

Let us consider how prototypes, concepts, and generalization hierarchy can emerge and be processed in distributed memories, such as the associative field of APNNs.

5.3.1 Correlated codevectors and emergence of generalization hierarchy

As discussed in sections 2 and 3, models of classes include some common sets of component models that allow ascribing them to a class. In APNNs, representations of the models consist from the reduced distributed representations of their component models. (Reduction by the Context-Dependent Thinning procedure is described in section 6). Since the reduced distributed representations of similar composite items are similar, the codevectors of instance items from a class are similar, i.e., correlated as well.

Thus, under learning in the associative field with gradual or stochastic binary learning rule, frequent combinations of 1s in input codevectors form the core parts of assemblies - the parts with increased connectivity. Less common combinations of active neurons form assembly fringes. Therefore a rather complex and rich structure of assemblies emerges in the assembly neural network, where one can discover "strong" cores with strong connectivity, "weak" cores with less strong connectivity, and all "intermediate" classes of cores.

The assembly cores of different strength represent the regularities (components and their combinations) that integrate objects into categories, thus corresponding to concepts. In assembly fringes, representations of components or attributes peculiar to individual items (class instances) are fixed.

Thus, a flexible classification structure corresponding to the category-based or generalization hierarchy is realized in the internal structure of assemblies formed in the assembly neural network. Such a complex inner structure of assemblies corresponding to classification hierarchy of taxonomy type occurs naturally, without introduction of any artificial classification.

The cores smaller in size and with stronger interconnections correspond to more general classes. This allows abstractness or generality level of representations to be dynamically controlled during the network functioning by setting the level of activity with the threshold value. An assembly in its extended representation corresponds to an instance; a narrower representation corresponds to a sub-class, a still narrower core corresponds to a class, etc.

"Travelling" in the hierarchy tree can be performed by a smooth change of the network activity with an external activity control system (see also Tsodyks 1990). In combination with changing the attributes at the input, threshold control allows context to be taken into account in the process of an assembly activation (Figure 5.3). For example, the resulting activity may be a "context-dependent chunk" - a part of assembly corresponding to few attributes from their total set for the item, or various assembly cores corresponding to various classes of the input probe. Actually, the complex internal structure of a neural assembly allows building a virtually continuous variety of hierarchical transitions.

Adding the representation of a name (label, word) attribute to some assembly core corresponds to formation of a concept. Name assigning allows the use of language for activation of assemblies. Naming allows some gradations of the categorization hierarchy to be identified. Also, naming provides an easier distinguishing between concepts (Harnad 1987; Cangelosi & Harnad 2000).

Let us consider an example of formation of object assemblies in the associative field of visual modality. Let us describe such objects as a field covered with grass, a forest edge, a free standing tree, a rushy swamp, etc. The attributes of color, texture, shape, size, position, and other attributes are extracted from the visual sensory information and their models are encoded by the distributed codevectors. The objects mentioned above will have a number of common sensory attributes (the main color attribute is green, the basic texture attribute contains a lot of micro-edges, the main dynamic property is small movements of edges forced by wind, etc.).

In the process of assembly formation, representations of the attributes that are met in all objects will form the assembly core. Its connectivity will be stronger than the connectivity of the assembly parts representing the attributes peculiar to individual objects. Green color and a lot of micro-edges can comprise a rather pronounced core for vegetation. If we consider more attributes, e.g., shape, it is possible to distinguish subclasses. E.g., the core of a "tree" subclass incorporates the "vegetation" class core, but the neurons of subclass are on the average less connected with each other than the neurons of class. By extending the number of active neurons in the considered assembly (and therefore the attribute set), it is possible to come down to the representations of specific objects.

If we need to operate with such general or abstract representations as a stone or vegetation, the level of activity should be made small by setting a high threshold. If we wish to know what precisely is in front of us, a bush or a bunch of grass, the activity controller should allow more neurons from the assembly fringes to be activated, to add specific attributes to the concept.

5.3.2. Correlated patterns and prototypes in distributed memories: related work

The processes of storage and retrieval of correlated patterns in distributed memories have been studied by a number of researchers.

Research of hierarchically correlated patterns in distributed memory has been initiated by physicists who studied the "ultrametric" organization of ground states in spin-glasses and emergence of spurious memories in Hopfield network (e.g., Hopfield, Feinstein, & Palmer 1983; Amit 1989; Parga & Virasoro 1986; Feigelman & Ioffe 1987; Gutfreund 1988; Tsodyks 1990; Kakeya & Okabe 1999; Hirahara, Oka, & Kindo 2000).

Vedenov (1988) considered emergence of spurious attractors in Hopfield networks as a side effect of the main function of distributed associative memory consisting not in memorizing individual patterns, but in formation of prototypes (or assembly cores in our terms). A large overlap of random dense codevectors used in Hopfield networks increases the probability of emergence of random sets of neurons with high connectivity that form spurious attractors. Such an interpretation is close to the earlier work of J.A. Anderson (Anderson et.al 1977; Anderson 1983, Anderson & Murphy 1986) and the PDP group (McClelland & Rumelhart 1986) who considered formation of concepts, prototypes, taxonomic hierarchy as a natural generalization of correlated patterns memorized in a distributed network. Tsodyks (1990) theoretically investigated the networks with sparse binary hierarchically correlated patterns, the "pure Hebbian" learning rule for gradual connections (equation 5.4), "natural" formation of cores and fringes, and threshold control to walk through the hierarchical generalization tree.

Baidyk & Kussul (1992) made a preliminary experimental study of complex internal structure of assemblies formed from sparse binary correlated codevectors with the stochastic version of the Hebbian learning rule (equation 5.5). They have simulated formation of the assemblies with cores and fringes to investigate the storage capacity of the APNN associative field. The network of 4096 neurons was used, and assemblies with approximately 120-200 neurons (about 60 neurons in the core part and 60-140 neurons in the fringe) were formed. Special tests have been performed, such as retrieval of a core by its part, retrieval of a core and its full fringe by the core and a part of fringe, retrieval of a core by a part of its fringe. The most difficult test was found to be retrieval of a fringe provided with its core and a part of that fringe. As expected, the experiments have shown a substantial decrease of storage capacity compared to statistically independent codevectors. A special learning rule was proposed to increase the stability of fringes.

Recently, the work on heterogeneous assemblies has been continued in the problem of handwritten word recognition (Kussul & Kasatkina, 1999), where an assembly neural network was used to accumulate the frequency of letter triplet occurrence in words. Further investigation of generalization hierarchy emergence and processing of hierarchically correlated patterns in assembly neural networks is required, and some problems to be solved are discussed in section 10.

6. Binding and making reduced representations by the Context-Dependent Thinning

In APNNs, binding is realized by the Context-Dependent Thinning (CDT) procedure. Various versions of the CDT (or thinning) procedure are discussed in Rachkovskij & Kussul (2001).

6.1. The subtractive CDT procedure

Here we will describe the "subtractive" version of the CDT procedure (Rachkovskij & Kussul 2001). This procedure was originally proposed by Kussul under the name "normalization" (Kussul 1988; Kussul & Baidyk 1990; Amosov et al. 1991; Kussul1992).

A single codevector **Z** is input to the procedure. This codevector is the disjunctive superposition of two or several component codevectors to be bound:

$$\mathbf{Z} = \bigvee_{s=1}^{S} \mathbf{X}_s, \quad (6.1)$$

where S is the number of components, $\mathbf{X}_s$ is the (randomly generated) codevector of the s-th component.

Let us mask **Z** with the inverse of disjunction of permuted versions of **Z**:

$$\langle \mathbf{Z} \rangle = \mathbf{Z} \wedge \neg \left(\bigvee_{k=1}^{K} \tilde{\mathbf{Z}}(k) \right) = \mathbf{Z} \wedge \left(\bigwedge_{k=1}^{K} \neg \tilde{\mathbf{Z}}(k) \right). \quad (6.2)$$

Here, $\tilde{\mathbf{Z}}(k)$ is **Z** with permuted elements, $\neg$ is the bit inversion, $\langle \mathbf{Z} \rangle$ is the thinned codevector. Each k-th permutation must be fixed, unique, and independent. Random permutations would be ideal, however cyclic shift permutations with a random number of shifts are convenient to use in implementations.

The CDT procedure reduces the density of the input codevector **Z** down to some specified value. Usually it is comparable to the density of its component codevectors $\mathbf{X}_s$. The number K of permuted and disjunctively superimposed vectors is chosen so that the number of 1s in $\langle \mathbf{Z} \rangle$ becomes approximately as needed. As shown, e.g., in Rachkovskij & Kussul (2001)

$$K \approx \ln S / pS, \quad (6.3)$$

where p is the density of 1s in the component codevector (and approximately equal to the density of 1s in the resulting thinned codevector).

Let us consider a toy example of thinning (Figure 6.1). Two component codevectors $\mathbf{X}_1$ and $\mathbf{X}_2$ each has m=2 of 1s in total of n=12 bits. (Remember, that actually M≈1000, N≈100,000). $\mathbf{Z} = \mathbf{X}_1 \vee \mathbf{X}_2$ has 4 of 1s. It's first permutation $\tilde{\mathbf{Z}}(1)$ is obtained as 4-bit down shift. It can be seen that $\mathbf{Z} \wedge \neg\tilde{\mathbf{Z}}(1)$ eliminates single 1 in **Z**. $\mathbf{Z} \wedge \neg\tilde{\mathbf{Z}}(1) \wedge \neg\tilde{\mathbf{Z}}(2)$, where $\tilde{\mathbf{Z}}(2)$ is 1-bit down shift of **Z**, eliminates no additional 1s. However, after conjuction of the result with the inverse of $\tilde{\mathbf{Z}}(3)$ (obtained as 2-bit down shift of **Z**), m=2 of 1s remain in $\langle \mathbf{Z} \rangle$, and we stop the thinning procedure.

6.2. A neural network implementation

Let us consider a neural network implementation of this procedure (Figure 6.3). There are three buffer fields of the same number of neurons: two input fields F_{in1} and F_{in2}, as well as the output field F_{out}. The copy of the input vector **Z** is in both input fields. The neurons of F_{in1} and F_{out} are connected by the bundle of direct projective connections (1-to-1). The neurons of F_{in2} and F_{out} are connected by K bundles of independent permutive connections. The synapses of permutive connections are inhibitory (the weight is -1). The threshold of the output field neurons is 0.5. Therefore the neurons of **Z** remaining active in F_{out} are those for which none of the permutive connections coming from **Z** are active.

Since the value of $\ln(S)/S$ is approximately the same at S=2,3,4,5 (Table 6.1), one can choose the K for a specified density of component codevectors $p(\mathbf{X})$ as:

$$K \approx 0.34 / p(\mathbf{X}). \quad (6.4)$$

At such K and S, $p(\{\mathbf{Z}\}) \approx p(\mathbf{X})$. Therefore the number K of permutive connection bundles can be fixed for certain density of the component codevector and the thinned codevector. So each neuron of F_{out}

may be considered as connected to an average of K randomly chosen neurons of F_{in2} by inhibitory connections. More precise values of K for different values of p are presented in Table 6.1.

Table 6.1. The function ln$S$/S and the number K of permutations of an input codevector that produces the proper density of the thinned output codevector in the subtractive version of the Context-Dependent Thinning procedure. K should be rounded to the nearest integer.

	Number S of component codevectors in the input codevector					
	2	3	4	5	6	7
ln$S$/S	0.347	0.366	0.347	0.322	0.299	0.278
p=0.001	346.2	365.7	345.9	321.1	297.7	277.0
p=0.005	69.0	72.7	68.6	63.6	58.8	54.6
p=0.010	34.3	36.1	34.0	31.4	29.0	26.8

A lot of orthogonal "patterns" of thinning using different patterns of permutive connections are possible. We will denote different thinning patterns by different labels shown as the superscript of the left angle bracket, e.g., ${}^{1}\langle\mathbf{Z}\rangle$, ${}^{2}\langle\mathbf{Z}\rangle$,.${}^{5}\langle\mathbf{Z}\rangle$, ${}^{u}\langle\mathbf{Z}\rangle$, where *1*, *2*, *5*, *u* are the labels denoting different thinning patterns. Inside each module of APNNs, the thinning pattern is permanent.

6.3. Preservation of similarity by the CDT procedure

Thinning in APNNs preserves both unstructured and structured similarity of the codes.

Thinning preserves the 1s of the input codevector **Z** (Eq. 6.2) with equal probability. Therefore, the thinned codevector is similar to the codevectors of all components superimposed in **Z**. The similarity of the resulting codevector to its component codevectors is called *unstructured similarity* (Plate 1997).

The fraction of 1s in the output thinned codevector from each of component codevectors is proportional to the density of the component codevectors. For example, for independent **A**,**B**,**C** of the same density,
$\langle\mathbf{Z}\rangle = \langle\mathbf{A} \vee \mathbf{B} \vee \mathbf{C}\rangle$; $p(\langle\mathbf{Z}\rangle \wedge \mathbf{A}) = p(\langle\mathbf{Z}\rangle \wedge \mathbf{B}) = p(\langle\mathbf{Z}\rangle \wedge \mathbf{C}) = p(\langle\mathbf{Z}\rangle)/3$.
If $p(\langle\mathbf{Z}\rangle) = p(\mathbf{A}) = p(\mathbf{B}) = p(\mathbf{C})$, approximately 0.33 of 1s will remain from each of the component codevectors.

However, the subset of 1s preserved in the thinned codevector from each component depends on the other components in the input superposition. For the example of three-component items, the subset of 1s from **A** preserved in $\langle\mathbf{A} \vee \mathbf{B} \vee \mathbf{C}\rangle$ is different from the subset of 1s from **A** preserved in $\langle\mathbf{A} \vee \mathbf{B} \vee \mathbf{D}\rangle$ and $\langle\mathbf{A} \vee \mathbf{D} \vee \mathbf{E}\rangle$ (Figure 6.4). Therefore, thinned code is bound - representation of 1s from each component depends on the other components present.

Similar component sets produce similar thinned codes, preserving *structured similarity*. The character of structured similarity is shown in Figure 6.5. The input vector was superposition of five independent component codevectors of the same density. The similarity value of two input vectors was varied by changing the composition of components. For example, if we choose the first codevector as **Z1** = $\mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \vee \mathbf{D} \vee \mathbf{E}$ and the second one as **Z2** = $\mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \vee \mathbf{F} \vee \mathbf{H}$, their similarity (overlap) value will be approximately 0.6. Depending on the density of the thinned codevectors, their similarity varies from linear (additive similarity of superposition corresponding to the "shallow" thinning) to approximately cubic (similarity value of about 0.55 for 4M versus about 0.18 for M/4).

6.4. Auto-thinning and hetero-thinning

So far we considered the version of the thinning procedure where a single vector (superposition of component codevectors) was the input. The corresponding pattern of activity was present both in the field F_{in1} and F_{in2} (Figure 6.3), and, therefore, the input vector thinned itself. This version of thinning can be also called “auto-thinning”. Unless otherwise specified, it is assumed that the number K of bundles is chosen to maintain the preset density of the thinned vector $\langle\mathbf{U}\rangle$, usually $|\langle\mathbf{U}\rangle| \approx M$.

It is possible to thin one codevector by another one if the pattern to be thinned is activated in F_{in1}, and the pattern which thins is activated in F_{in2}. Let us call such procedure hetero-CDT, hetero-thinning, thinning **U** with **W**. We denote hetero-thinning as

$^{\text{label}}\langle \mathbf{U} \rangle_{\mathbf{W}}$. (6.5)

Here **W** is the pattern that does thinning. It is activated in F_{in2} of Figure 6.3. **U** is the pattern which is thinned, it is activated in F_{in1}. $^{\text{label}}\langle \ldots \rangle$ is the configuration label of thinning. For auto-thinning, we may write $\langle \mathbf{U} \rangle = \langle \mathbf{U} \rangle_{\mathbf{U}}$.

For the subtractive hetero-thinning, equation 6.1 can be rewritten as

$$\langle \mathbf{U} \rangle_{\mathbf{W}} = \mathbf{U} \wedge \neg (\bigvee_{k=1}^{K} \mathbf{W}^{\sim}(k)). \quad (6.6)$$

We have used both auto-thinning and hetero-thinning in the multilevel APNNs for sequence processing (Rachkovskij 1990b; Kussul & Rachkovskij 1991), see also section 8. Auto-thinning has been also used for binding of sparse codes in perceptron-like classifiers (Kussul, Baidyk, Lukovich, & Rachkovskij 1993), and in the one-level APNNs applied for recognition of vowels (Rachkovskij & Fedoseyeva 1990; 1991), word roots (Fedoseyeva 1992), textures (Artykutsa et al. 1991; Kussul, Rachkovskij, & Baidyk 1991b), shapes (Kussul & Baidyk 1990), handprinted characters (Lavrenyuk 1995), handwritten digits, letters, and words (unpublished work of 1991-1992 in collaboration with WACOM, Co., Japan, see also Kussul & Kasatkina 1999), logical inference (Kussul 1992; Kasatkin & Kasatkina 1991), analogical retrieval and mapping (Rachkovskij 2001).

7. A simple APNN structure - basic operations and examples

Let us consider basic processes in simple two-level APNN architectures.

7.1. A simple parallel scheme of APNNs

Consider two-level APNNs, where level 1 consists of two modules, and level 2 consists of one module (Figure 7.1). Each module comprises an associative field and several buffer fields connected by the bundles of projective connections. In the buffer fields, temporal storage of codevectors, their thinning, and various bit-wise operations are performed. Such a simple structure demonstrates the basic processes with compositional items in APNNs.

Let a composite item consist of two components represented by codevectors **A** and **B**. These components can be composite items in their turn. **A** and **B** are memorized in the modules M11 and M12 correspondingly using long-term memory of the associative fields or working memory of the buffer fields.

By projective connections, **A** and **B** are transferred to the input buffer field of M2, where they are superimposed by disjunction: **A** v **B**. Then this superposition is thinned, providing reduced and bound codevector <**A** v **B**>. (Figure 7.2). In the thinned composite codevector, only a fraction of 1s (active neurons) is preserved from **A** and **B**. Thus a reduced representation of the composite item is ***formed*** , where its components **A** and **B** are bound.

By projective connections, this reduced representation **<A v B>** can be transferred from the buffer field to the associative field of M2 and ***stored*** there by equation (5.5). This codevector can also be used as a probe to ***retrieve*** the "closest match" codevector from the associative field of M2. The closest match codevector could be, for example, **<A' v B'>**, where **A'** is similar to **A**, **B'** is similar to **B** (Figure 7.3). The content of ***similarity*** of composite reduced codevectors can be found as their overlap <**AvB**>&<**A'vB'**>, their symmetric ***difference*** is <**AvB**>^<**A'vB'**>, and an asymmetric difference is <**AvB**>&~<**A'vB'**>.

Let us consider ***decoding*** of a reduced composite codevector, i.e., retrieval of its component codevectors in full size. Transfer <**A v B**> by projective connections from M2 to M11 and M12. Use <**AvB**> as a probe to retrieve the "closest match" codevectors in both modules (by associative recall in the associative field or by direct comparison with the codevectors preserved in buffer fields). Since thinning preserves unstructured similarity, the reduced representation of a composite item is similar to its component representations, and <**AvB**> is similar to **A** and **B**. Therefore, **A** will be retrieved in M11, and **B** will be retrieved in M12. We assume, that there is no **B** in M11 and there is no **A** in M12.

Let us consider ***access to a whole*** **by its part** - to a composite codevector by its component codevector. If we have **A** in M11 and no codevector in M12, transfer **A** to M2 and use as a probe. If there is no more reduced composite codevectors with component **A** in M2, <**AvB**> will be retrieved there (Figure 7.6).

Let us consider an ***association*** between component codevectors. In our terms, association between **A** from M11 and **B** from M12 is realized by their binding (or reduced representation of association) <**AvB**> in M2 (Figure 7.7). If **A** is in M11, it will retrieve <**A v B**> as the whole by its part. After transferring <**A v B**> to M12, **B** will be retrieved there, which is the associated codevector. In the same way, **A** can be retrieved by **B**.

7.2. A simple serial scheme of APNNs

To process the composite items consisting of component codevectors from a single module, a serial APNN scheme should be used (Figure 7.8).

Let us consider ***formation*** of the reduced representation of a composite item. Components **A**, **B**, **C** are stored in M1. They are sequentially activated in the associative field and transferred by projective connections to the input buffer field of M2. There, they are superimposed and thinned, forming reduced representation of the composite item <**AvBvC**>. It can be ***stored*** in the associative field of M2 or/and be used as a probe for ***retrieval*** from memory. If the probe represents partial set of components of the stored composite codevector (e.g., **A** or **AvC**), ***then access to a whole by its part*** occurs.

Decoding of the reduced representation of the whole through the full-sized representations of its parts is realized sequentially. For this purpose, let us transfer <**AvBvC**> from the higher level module M2 via projective connections to the lower level module M1, preserving it in M2 as well. Since this reduced representation of the whole contains the portions of its component representations, the

assembly of some component will be activated in the associative field of M1. It can be the component assembly with the largest representation in the composite pattern, or with the strongest connectivity, or the component assembly that won the competition for activity for some other reason. Let it be **C**. To retrieve the other components, let the control system inhibit the neurons representing the already retrieved component (**C** in our case) in the reduced compositional pattern remaining in the buffer field of M2: <**A**v**B**v**C**>&~**C.** Now, if we transfer the remaining part of the thinned codevector from M2 to M1, some other component assembly will be activated. We may continue this process until there remain no active neurons in the buffer field of the higher level module, thus retrieving all component codevectors (Figure 7.9).

The combination of processes of the whole formation, access to the whole by its part, and decoding of the whole, may be used to ***associate*** items in the same manner as for the parallel scheme of the previous section. In this case, known items (e.g., **A**) are first eliminated from the reduced representation of the association (e.g., <**A**v**B**v**C**>&~**A**).

7.3. Interpretations of the processes in the APNN architecture

Depending on semantics of the codevectors represented in the APNN modules, various interpretations may be given to the semantics of the reduced composite codevector and to the processes in the APNN architecture considered in previous sections 7.1-7.2.

For example, the reduced representation of the composite item consisting of component representations may be treated as the representation of:
- an object-whole (real or abstract) consisting of objects-parts;
- an object described by its global attributes;
- a binding (e.g., variable-value, role-filler);
- an undirected relation between objects;
- an undirected association between arbitrary items (objects, an object and its name, an object and response, a situation and its evaluation, an action and the object changes it produces, conditions and inference, cause and effect, antecendent and consequent, etc.);
- and so on.

The operations considered and their interpretation can be naturally extended to the APNN architecture with a larger number of levels and modules in a level, allowing representations of complex compositional items-models to be processed. The number of components forming the chunk at each hierarchical level is somewhat restricted by the capability of associative fields to retrieve reliably component representations from the reduced representation of the composite.

Representations and architectures for processing of ordered items are considered in section 8.

7.4 Example 1. Propositions with undirected relations and analogies

Propositions with undirected relations are considered as composite items, where the arguments (objects) and predicate (relation) are the components. For example, the relational instance *John and Mary are in love* may be represented as

R1=<**John** v **Mary** v **in_love**>.

John and **Mary** may be represented and memorized in the same module (e.g., in M11 of Figure 7.1), and **in_love** - in another module (e.g., in M12). The reduced representation **R1** of the relational instance is formed and stored in M2.

R1 may be retrieved in M2 by a partial set of its components, and missing components may be decoded in the lower-level modules. For example, the assembly of **R1** will be retrieved in M2 by the probe <**John** v **in_love**>. Decoding of **R1** & ~**John** in M11 will retrieve **Mary,** since it does not contain reduced representation of **John**.

Let us form in M2 the reduced representation **R2** of the relational instance *"John and Mary will marry":*

R2 = <**John** v **will_marry** v **Mary**>.

R2 can be retrieved in M2 by the probe **R1**, providing us with the associated information about John and Mary. (Alternatively, such an association could be established in the still higher-level module M3: <**R1**v**R2**>).

John and **Mary** in M11 are complex composite codevectors composed from the reduced representations of their attribute descriptions. As an illustration, let us assume that the context has chosen the following attributes to be chunked in the representation of John and Mary:

John = <**human** v **male** v **John_name**>,
Mary = <**human** v **female** v **Mary_name**>.

Then the representation of relational instance *"John and Mary are in love"* will be similar to the representation of such relational instances as *"Bill and Kate like each other":*

R3 = <**Bill** v **Kate** v **like_each_other**>,

especially if **like_each_other** and **in_love** are chosen to be similar. This allows **R1** to be retrieved in M2 given **R3** as the probe, providing the simplest example of *analogical access*. By decoding and comparing the component codevectors representing the elements of these relational instances, the following correspondences may be established:

Bill <-> *John*,
Kate <-> *Mary*,
like each other <-> *are in love*,

providing the simplest example *of analogical mapping*. Then the information about possible future evolution of interrelations between Bill and Kate can be inferred. Retrieving **R2** by **R1**, and using the mapping above, we can predict that *"Bill and Kate will marry"*. This is a simple example of *analogical transfer*. Probably, this prediction would be more reliable if **R1** and **R3** had a higher similarity - e.g., if **R2** had relation (predicate) *"are in love"* instead of *"like each other"*.

7.5. Example 2. A multi-modal sensory model of an object

Let us consider an example of the APNN architecture for sensory representation of objects. This is a simplified scheme, where an object is represented only by its global attributes, without taking into account its parts and their relationships, dynamic attributes, and other possible attributes.

This architecture includes the modules of independent modalities and sub-modalities (Figure 7.10). For example, such independent visual attributes of an object as shape, size, texture, color are represented in the modules M11...M14. Visual model of an object is formed from these attributes in the module M21. In the modules M22...M25 of this compositional level, acoustical, tactile, olfactory, taste models are formed. Integral multi-modality sensory model of an object is formed in the module M31, to which name attribute could be added from the module M32. Probably, name could be an attribute in the models of all modules.

Such a scheme allows a fast parallel decoding of the sensory attributes of an object. After identification of an object by its representation in the module of some modality, an associated information can be retrieved, such as sensory representation of other modalities and sub-modalities, or the name. In the same manner, sensory representation of objects in different modules and modalities can be retrieved by its name representation.

8. Processing of ordered items in APNNs

For items considered in the previous section, the order was not important, or explicitly prescribed by their module. For example, in the parallel scheme of section 7.1, the representation of a specific item can only be stored in the unique module, the items from different modules are represented by uncorrelated codevectors, the items from one module always precede the items from the other module. However, in many cases the order of items from the same module is important. For example, many relations are directed, such as spatial ones (mutual positioning), relations that take time into account (e.g., cause-effect, situation1-action-situation2), a more abstract sequences (AB vs BA), and so on.

The order or sequence may be represented by some kind of binding an item codevector with its place in a sequence (Kussul & Rachkovskij 1990; Rachkovskij 1990b; Plate 1995; Rachkovskij & Kussul 2001). One technique consists in binding an item codevector with codevectors of the preceding items and their subsequences that provide the context indicating the item's place in the sequence. Another technique consists in binding with the representation of the item's ordinal number in the sequence. Let us consider three versions of the second technique: binding an item codevector with the codevector encoding its place in the sequence using auto-thinning; using hetero-thinning; and implicit binding with the place by modifying the item codevector without thinning.

Since the number of items in a sequence can be large, the sequence must be chunked and represented hierarchically. For example, a sequence (chunk) of letters forms a syllable, a sequence of syllables forms a word, a sequence of words forms word combination, a sequence of word combinations forms a sentence, etc.

8.1. Representation of sequence using auto-thinning with the number

To represent a sequence, role-filler bindings can be used. The codevectors of the items from the sequence fragment (chunk) are sequentially activated in M1 (Figure 8.1). In M#, the codevectors of their ordinal numbers are activated. The combination of two codevectors - the item from the chunk and its number - are superimposed, thinned, and stored in M2, e.g., ^{1}<**A** v **#1**>, ^{1}<**B** v **#2**>, ^{1}<**C** v **#3**>. These bindings are also transferred to M3, accumulated by disjunction, and thinned there, forming the reduced representation of the sequence fragment, e.g., ^{2}<^{1}<**A** v **#1**> v ^{1}<**B** v **#2**> v ^{1}<**C** v **#3**>>.

To decode the sequence fragment, its reduced representation is transferred from M3 to M2. At the same time, **#1** is activated in M#, transferred to M2, and superimposed with the chunk codevector: ^{2}<^{1}<**A** v **#1**> v ^{1}<**B** v **#2**> v ^{1}<**C** v **#3**>> v **#1**. As a result, the first binding of the fragment, ^{1}<**A** v **#1**>, will have more 1s remained from its full representation in M2 (because of the 1s from **#1**), and will be retrieved there. Then, it can be transferred to M1, where the first codevector of the chunk, **A**, will be retrieved. In the same way, the second and the third items of the fragment will be retrieved. The number of items in the fragment can be determined by transferring its codevector ^{2}<^{1}<**A** v **#1**> v ^{1}<**B** v **#2**> v ^{1}<**C** v **#3**>> to M# and finding the most similar codevectors of the ordinal numbers there.

In order to check that the chunk codevector is correctly decoded, the sequence representation of the decoded components can be formed in M3 and compared to the initial chunk codevector. Also, the decoded bindings can be eliminated from the chunk codevector as in section 7.2, e.g., ^{2}<^{1}<**A** v **#1**> v ^{1}<**B** v **#2**> v ^{1}<**C** v **#3**>> & ~^{1}<**A** v **#1**>, etc.

8.2. Representation of sequence using hetero-thinning with the number

In the scheme of sequence processing using hetero-thinning with the role, the item codevectors are stored in M1 and the codevectors of ordinal numbers are stored is M#. To represent the item of a certain ordinal position in a sequence chunk, its codevector is hetero-thinned (section 6.4) with the codevector of the ordinal number: <**X**>$_{\#}$, e.g., <**A**>$_{\#1}$. Such hetero-thinned representations of all the components of a sequence fragment are disjunctively accumulated in M2 and auto-thinned there, e.g., ^{2}<^{1}<**A**>$_{\#1}$ v ^{1}<**B**>$_{\#2}$ v ^{1}<**C**>$_{\#3}$> (Figure 8.2)**.**

For decoding, the reduced representation of a sequence fragment is transferred from M2 to M1 and is hetero-thinned from **#1**: ^{1}<^{2}<^{1}<**A**>$_{\#1}$ v ^{1}<**B**>$_{\#2}$ v ^{1}<**C**>$_{\#3}$>>$_{\#1}$. Since the second thinning with the same codevector and the same pattern of thinning connections does not reduce the number of 1s in the thinned codevector, the number of 1s from **A** will be greater than the number of 1s from **B** and **C**. Thus **A** will be found as the closest match in M1. In the same way, hetero-thinning of the reduced representation of the sequence chunk with the codevectors of other ordinal numbers (**#2**, **#3**, etc.) will retrieve in M1 the other sequence items in correct order (**B**, **C**, etc.).

8.3. Representation of sequence by the place of code

One of the well-known techniques to represent the ordinal number of an item is to allocate a separate pool of units for each possible position of an item in a sequence chunk. Such a technique leads to the growth of code dimensionality. To preserve fixed dimensionality of codevectors, it was proposed to encode different positions of an item by a (cyclic) shift (Kussul & Baidyk 1993a, Sjodin 1998) or other invertible permutation (Rachkovskij & Kussul 2001) of the item codevector.

In such a scheme, sequence processing may look as follows (Figure 8.3). Each item codevector from M1 undergoes a certain shift or permutation, depending on its position in a sequence fragment. Permuted codevectors are disjunctively superimposed and thinned in M2, forming reduced representation of the sequence chunk:

$^{1}<$ (**A**>>#1) v (**B**>>#2) v (**C**>>#3) >,

where (**X**>>#n) stands for **X** permuted or shifted to encode its n-th position in a sequence chunk. Decoding is done in the following way. The reduced representation of a sequence chunk is subjected to the inverse of the shift or permutation that was used to encode the first position of the item codevector:

$^{1}<$(**A**>>#1) v (**B**>>#2) v (**C**>>#3)> <<#1.

X<<#n stands for the inverse of the shift or permutation that encode the n-th position of an item in its chunk. Then this codevector is transferred to M1, where **A** is retrieved. In the analogous way, after inverse of transformations >>#2 and >>#3, **B** and **C** are retrieved in M1.

Note, that in this scheme, unlike two previous ones, representations of sequence chunks consisting of the same items in different order are not similar. For example, codevector of *XYZ* is unsimilar to codevector of *YZX*. In order to preserve the similarity, non-permuted codevectors should be also included into reduced representation of the chunk (Rachkovskij 2001, see also section 8.5).

8.4. Example 1. An APNN architecture for processing of letter sequences

Let us use the scheme of section 8.2 to process letter sequences. The corresponding APNN structure may look as in Figure 8.4. The codevectors of letters, syllables, words, and their ordinal numbers are formed, stored, and processed in the corresponding modules. For example, in order to process the word *SEQUENCE*, it is segmented into fragments or chunks corresponding to syllables: *SE-QUE-NCE*. Then it can be represented as

$^{4}<^{3}<^{2}<^{1}<$**S**$>_{\#11}$ v $^{1}<$**E**$>_{\#12}>>_{\#21}>$ v $^{3}<^{2}<^{1}<$**Q**$>_{\#11}$ v $^{1}<$**U**$>_{\#12}$ v $^{1}<$**E**$>_{\#13}>>_{\#22}>$ v $^{3}<^{2}<^{1}<$**N**$>_{\#11}$ v $^{1}<$**C**$>_{\#12}$ v $^{1}<$**E**$>_{\#13}>>_{\#23}>>$.

Processing of memorized words (recognition, decoding through letters) is performed as in section 8.2. The APNN architectures and mechanisms of memorizing, recognition and replaying of complex letter sequences of various length, up to word sequences, including branches and repetitions, were examined in Rachkovskij (1990b), Kussul & Rachkovskij (1991).

8.5. Example 2: Propositions with directed relations

Let us consider representation of relational structures, for example, the representation of "*Spot bit Jane, causing Jane to flee from Spot*". Using APNN representations with role-filler bindings, it may look as (Rachkovskij 2001):
P_apnn(bite) = $^{2}<^{1}<$**bite_agt** v **spot**> v $^{1}<$**bite_obj** v **jane**>>,
P_apnn(flee) = $^{2}<^{1}<$**flee_agt** v **jane**> v $^{2}<$**flee_obj** v **spot**>>,
P_apnn = $^{4}<^{3}<$**cause_antc** v **P_apnn**(bite)> v $^{3}<$**cause_cnsq** v **P_apnn**(flee)>>.

Using shift (or other invertible permutation) of codevectors to encode the order, representation can be as follows:
P_apnn1(bite) =
$^{1}<$**P_apnn1-structure**(spot-jane) v **P_apnn1-components**(bite,spot,jane)>,

where
P_apnn1-structure(spot-jane) = **spot**>>agent v **jane**>>object,
P_apnn1-components(bite,spot,jane) = **bite** v **spot** v **jane**.

As mentioned in section 8.3, since
P_apnn1-structure(spot-jane) = **spot**>>agent v **jane**>>object
is not similar at all to
P_apnn1-structure(jane-spot) = **jane**>>agent v **spot**>>object,
it should be augmented with the component representation.

P_apnn1(flee) is constructed in the same manner:
P_apnn1(flee) =
^{1}<**P_apnn1-structure**(jane-spot) v **P_apnn1-components**(flee,jane,spot)>,
and the whole proposition looks as:
P_apnn1 = ^{2}<**cause** v **P_apnn1**(bite) v **P_apnn1**(flee)
v **P_apnn1**(bite)>>cause_antc v **P_apnn1**(flee)>>cause_cnsq>.

To build APNN according to that scheme, it is necessary to encode the entities and relations using base-level codevectors, by which we mean here independent stochastic codevectors of the lowest compositional level. In Rachkovskij (2001), the following representation was used (see also Plate 2000). All relations and roles were considered dissimilar and were represented by the base-level codevectors, e.g., **bite**, **bite_agt**, **bite_obj**, etc. The codevectors of objects (or tokens of some type) were formed as superposition of base-level codevectors of two attributes: that of type (**human**, **dog**) and that of identifier or name (**id_john**, **id_spot**, etc.). For example, the objects *John* and *Spot* were represented respectively as **john** = <**human**+**id_john**>, **spot** = <**dog**+**id_mort**>.

In Rachkovskij (2001), such representations were used to model the processes of human analogical access and mapping.

8.6. Example 3: Visual object recognition

Physical objects consisting of parts with various visual characteristics can be visually recognized by shape and mutual position of areas with different texture and color corresponding to their parts. Of many approaches to structural object recognition, as well as to representation and recognition of textures and shapes, most relevant are described in Kussul, Rachkovskij, & Baidyk (1991a), Kussul (1992), Hummel & Biederman (1992), see also Plate (1999). Extraction of global shape attributes and their use for image recognition was considered in Kussul & Baidyk (1990).

Our approach to texture representation and recognition is described in Kussul, Rachkovskij, & Baidyk (1991a,b), Amosov et.al. (1991), Kussul (1992), see also Goltsev (1996). We have used the following image characteristics inside an image window as textural attributes: the brightness histogram, the contrast histogram, the elementary edges' orientation histogram. Probably, one of the main functions of texture is providing segmentation of image regions with different visual characteristics (e.g., corresponding to different objects or their parts).

Various relations between image regions may be used for description and representation of an object's shape, such as spatial relations and relative sizes of image regions. For example, a tree trunk is usually located below and near a crown. The trunk area is considerably less than the crown area. The area, the centers of gravity, and the mutual location can be calculated for various image regions. A number of relations *R(A,B)* between two regions *A* and *B* can be determined, including undirected (horizontal, vertical, near, touch) and directed (left, right, above, below, larger, smaller). To describe an object, it is necessary to indicate specific regions that are characterized by these and other relations. Some of the mentioned relations can be characterized by numerical values. Encoding of numerical values in APNNs is described in Kussul (1992), Kussul, Rachkovskij, & Baidyk (1991a,b), Rachkovskij (1990b), see also section 10.1.

A toy example of APNNs for spatial relation encoding and structural object recognition is presented in Figure 8.5. The texture module T and the shape module S possess complex structure including several neural fields. The relation module R extracts spatial relations between uniformly textured regions. The position module P represents positions of the texture regions. The object name module NAME inputs and encodes the object's name.

Such a scheme operates in the following way. The image is fed to the T module, where uniformly textured regions and their texture type are determined. The "region map" of the current region is transferred to the S module, where its shape is represented and may be recognized, and to the P module, where its position is represented. The codevectors of texture, shape, and position of the

image region is transferred to the A1 module, where its reduced representation is formed by thinning and transferred to the A2 module for temporal storage.

Then another uniformly textured region neighboring the previous one is encoded in A1. The spatial relation between two regions is extracted in R. The codevectors of these two texture regions and their spatial relation are thinned and memorized in the higher-level module R-A1-A2. After looking through all texture regions of the object, the code in the R-A1-A2 module consists of a set of Relation-Area1-Area1 triplets. This code is superimposed with the object name code from NAME and thinned in the OBJECT module.

The procedure of object recognition is constructed in the same way, except for the object name that is unknown. After an object's assembly is activated in the OBJECT module, the name codevector is then decoded in NAME. The assembly activated in the associative field of OBJECT is considered as a hypothesis about the object in the field of vision. This hypothesis must be verified. For this purpose, the activated assembly must be decoded through the components of the lower-level modules. By comparing decoded representations of regions with the observable ones, one may discover that the decoded assembly contains unobservable regions. This allows initiating search of the missing texture regions on the basis of the decoded texture features as well as their supposed location relative to the already discovered texture regions.

Thus, object recognition requires iterative procedure including transition from the lower hierarchical levels to the higher ones as well as in the opposite direction. This procedure permits one to verify the recognition correctness, to specify the object and position of its parts, to tell the object from the background, to determine if it is shaded by another object, etc.

8.7. Example 4: Decision making and action planning - FA revisited

As mentioned in section 2.5, in the functional acts of interaction with the environment, the System is often required to make its own decisions and to plan actions. As an example of the System, we may consider an autonomous robot in complex environment about which the robot has no full a-priori information.

In the modules of higher levels, generalized assemblies of situations encountered by the robot are formed along with the comparative descriptions of the situations, where their similarities and differences are represented. The assemblies of rather complex sequences of actions for reaching certain goals are also formed there.

For example, the robot met an obstruction on the road and needs to strip it to move further. For this purpose, the robot has to perform a lot of actions. If the assembly corresponding to these actions and the assembly corresponding to the changes in the situation before and after obstruction dismantle have been already formed in their modules, then a new assembly can be built in another module which includes the reduced representation of executed actions and resulting change in the environment. Accumulation of assemblies associating the actions with the changes in the environment provides the associative basis for decision making and action planning. It should be noted, that the assemblies associating the robot actions with the resulting changes must form for the goals of various complexity - from the simplest (to make an elementary movement) to the most complex (to find a place in the forest that is convenient for robot parking).

In the modality of action assessment, there must be a module where the assemblies corresponding to the functioning criteria are formed. Some of those criteria may refer to the maintenance of serviceability and integrity of the robot. The input data for such assemblies are from sensors indicating the condition of robot's subsystems. However, there might also be assemblies which the developers of the robot introduced as criteria, and they may include components from any modality. Apparently, unsupervised learning should not be allowed in this module, otherwise it would mean that the robot will eventually begin to function in its own interests, whatever they may be.

Encountering various situations and executing various actions, the robot will concurrently read the excitation of assemblies-criteria that characterize assessment of situations and of robot's actions. The representations of changes of these estimations may be transferred to the modules where the assemblies describing the external and internal worlds of the robot are formed. Then representations of these changes may be included into all assemblies and characterize those assemblies with respect to improvement or deterioration in terms of the corresponding criteria. Such "evaluating" components are added to assemblies in much the same way as names. As a result, any process will be accompanied by the evaluation of results from the standpoint of the System criteria. Such a scheme is somewhat analogous to the emotional "coloring" (tinge) of information being processed.

In the modality of aim formation, a module is created where the assemblies corresponding to the desired combinations of criteria are formed. There may be rather stable criteria combinations, such as a rather large energy supply, serviceability of all subsystems, an accident-proof movement in the environment, etc. However, there may be temporal criteria prescribed according to the current task. Such assemblies may be input to the modality of aim formation through the speech modality. The assemblies corresponding to undesired combinations of criteria should be formed as well.

Formation of aims can be realized in autonomous and supervised modes. In the supervised mode, a user or another automatic system forms the desired combination of criteria in the corresponding module. Attaining a certain situation may be one of the criteria. This preset combination of criteria activates one of existing assemblies or forms a new one, thus providing the System with the aim - to reach the desired state of criteria.

In the autonomous mode, the main source of aims is the absence of undesired states. If the current estimate from the modality of action assessment activates some assembly of undesired combination of criteria, the aim is set for robot. If undesired combinations are absent, the assembly corresponding to the desired state closest to the current state becomes the aim.

After choosing the aim, the process of decision making is initiated. In the simplest case, it consists in finding the difference between the of current and target situations. Based on this difference, an action assembly associated with the elimination of the difference should be retrieved. Since associative recall of actions is done on the basis of approximate similarity, the anticipated situation to be achieved as a result of the chosen action might be not the target one. This predicted situation must be again compared to the target, and the process should be repeated. Such iterations should be done until small enough differences between the target and predicted situations are obtained, or until the intention to achieve the target situation is abandoned.

The aggregate of the obtained representations of actions leading to a good fit between target and predicted situations may be considered as an accepted decision or as a high-level plan. In order to make a more detailed plan, the assemblies of the higher levels corresponding to complex sets of actions should be decoded through the lower level assemblies corresponding to simple actions. During decoding, compatibility of actions with the environmental conditions should be taken into account. If incompatibility is discovered, the plan should be corrected.

9. Implementations of APNNs

As mentioned above, high-dimensional binary distributed representations of APNNs demand rather high volumes of computer memory for their simulation. It particularly concerns an associative field in the full-connected version. For example, for N=100,000, each buffer field requires 12.5 Kbyte, whereas a full-connected associative field requires 1.25 Gbyte of computer memory. Such a memory size is not common even for modern universal computers, and at the end of the 1980-s - beginning of the 1990-s it was practically unattainable. The following solutions for simulation and investigation of APNNs were proposed:

- Investigation of full-connected networks of smaller size (of the order of 4000 neurons and 16,000,000 connections);
- Use of hard-disk memory for storage of connection matrices;
- Use of not full-connected (diluted) networks. It allows a linear growth of memory for connections vs the number of neurons and corresponds to neurobiological data;
- Use of a common physical memory array for simulation of several associative fields;
- Modeling of separate functions of an associative field. For example, a non-distributed memory of size LxN bits in enough to retrieve the "closest match" codevectors. At $L<<N$, memory and speed gain may be substantial. Though such a memory organization allows various problems to be modeled (e.g., Plate 2000; Rachkovskij 2001), it does not permit investigation of natural emergence of category-based or conceptual hierarchy in distributed memory (section 5).

Binary codevectors and connections, modular structure of APNNs allowed their simple and high-performance implementation using specialized hardware - the Associative-Projective neurocomputer (APNC) (see, e.g., Kussul, Rachkovskij, & Baidyk 1991a; 1991b; Artykutsa, Baidyk, Kussul, E. M., & Rachkovskij 1991; Kussul, Lukovich, & Lutsenko 1988). The fact that most operations in APNNs are bit-wise facilitates the simplicity of the APNC implementations, and the sparseness of binary representations further accelerates the execution of many operations.

The APNC is a long-word processor that implements bit operations on the long words stored in a word-wide dynamic RAM. The number of simultaneously simulated neurons is equal to the number of bits in the processor word. The input sum I of each neuron (Eq. 5.1) is accumulated in a binary counter. In 1992-1993, several prototypes of APNCs were created together with WACOM Co., Japan as external modules for PC. PC provides the user interface and various auxiliary operations that require non-bit operations (integer and floating point arithmetic, etc). One version of the APNC had 512-bit words and memory size of 64 Mbyte, and another version had 2048-bit words and 256 Mbyte of memory. Since APNNs usually contain much more neurons than such a word length, the simulation process is only partially paralleled. One update cycle of the associative field of 10,000 neurons has taken several tens of milliseconds.

Similar hardware architecture was used in other neurocomputers simulating distributed associative memories (e.g., Palm 1993). At present, such memory size and performance are achievable with usual PC, providing favorable conditions for investigation of distributed representation usage in large-scale world models and applications based on distributed representations.

10. Discussion: some open issues and future directions

Though the paradigm of APNNs is under development since the early 1980-s, it remains open and calls for further investigation and elaboration.

Most of the tasks to which APNNs were applied used one- or two-level architectures (simple compositional items in terms of section 3) as classifiers (the mode that was not even considered in this article). Flat vectors of attributes were used as inputs. This work was reviewed in Kussul (1992, 1993). We have only used multi-level APNNs to demonstrate processing of abstract sequences (Rachkovskij 1990b; Kussul & Rachkovskij 1991), processing of texts (words and word combinations as sequences of letters, Rachkovskij 1996), and analogical access and mapping (Rachkovskij 2001).

The progress in personal computers provides the possibility to investigate various APNN mechanisms that have not been investigated so far and need further elaboration. Let us touch on some aspects that have to be solved for further development and applications of both APNN and other similar architectures, and on possible directions of future work in this area. They concern
- the real-world interface, i.e., the mechanisms of elementary attribute formation and representations, e.g., sensory input and motor output;
- representation and processing of hierarchies items - peculiarities of formation and retrieval of assemblies of various composition levels and generality;
- the specific APNN architectures for particular application problems.

10.1. Attributes

In APNNs, representations of item models are constructed from the representations of component models that are considered as the attributes of items. Some attributes may in turn have their own complex structure (see, e.g., Markman 1997). Therefore, using APNNs in realistic models of cognitive activity and in applications requires solution to the problems concerning which attributes may be considered elementary, how they can be extracted from the input information, how they can be distributedly represented with the proper regard to their similarity. These problems may be considered as specific versions of the well-known problems of meaning representation and symbol grounding (Hummel & Holyoak 1997; Kussul 1992; Plate 2000; see also Dorffner & Prem 1993; Barsalou 1999; Harnad 1990; Deerwester et. al. 1990). Then, the problem of which more complex attributes should be formed from the elementary ones by detection of changes or other relations between attributes must be solved as well. The problem of extraction of invariant sensory features is probably related to extraction of relations between certain attributes that remain constant for an object under various observation conditions.

To provide the System's interface with the external world, it is necessary to encode numerical values of certain attributes. In APNNs, procedures were proposed for encoding numbers and vectors of numbers by stochastic sparse distributed codevectors that reflect the metrics of the encoded parameters (Kussul 1992; Rachkovskij 1990b; Kussul, Rachkovskij, & Baidyk 1991a, 1991b; Kussul, Baidyk, Lukovich, & Rachkovskij 1994; Kussul, Rachkovskij, & Wunsch 1999). These procedures were used in our work on applications.

10.2. Associations and memory

In APNNs, various kinds of associations, such as associations between the items similar in terms of certain attributes (***A***<->***A'***), between a part and a whole (***A***<-> <***AB***>), between the parts of a whole (***A***<-><***AB***><->***B***), are searched through the similarity of representations. Associations and similarity of representations are context-dependent, therefore some part or subset of an item representation may be selected or get some advantage depending on the context. For example, when the reduced representation of a sequence fragment is decoded, the context provided by the representation of the component number selects the proper order of component retrieval (section 8).

To store and find various kinds of associations, we used an auto-associative memory just in order to demonstrate its capabilities. Traditionally, associations are implemented using a hetero-associative memory. Probably, hetero-associative memory can also be used in the APNN-based implementations of the world model, for example, its version with an enhanced information capacity recently investigated by Sommer & Palm (1999).

10.3. Context and control of associations

The influence of context in APNNs manifests itself in a variety of fashions. For example, the same probe codevector, depending on the direction of an association, may activate an assembly by its part, or a part of an assembly, or an assembly of the same complexity as the probe. Besides, depending on the context, the same probe may activate various assemblies of parts, wholes, or of the same complexity, or assemblies of various degree of generality, or various subsets of the same assembly. Since different subsets of an assembly correspond to different components of a concept, the content of a concept becomes dependent on the context.

The mechanisms of taking into account the context influence on the composition of an activated assembly and therefore on making associations need further investigation. This study should include the peculiarities of activating assemblies with a complex internal structure, which depend on:
- the relationship between the fractions of external and internal assembly activation;
- the activity level control;
- the history of assembly activation that influences current activation of neurons and their fatigue;
- the heterogeneity of an assembly structure.

Besides, the mechanisms need to be established that control
- transfer of representations between the modules,
- selection of external activations from different fields of the same and various modules,
- masking or filtering certain subsets of a probe or enhancing the other (e.g., selecting a particular set of attributes to form a chunk).

10.4. The part-whole hierarchy

In the described version of APNNs, parts and wholes are allocated to different modules. The reason is to be able to control the direction of movement through the part-whole hierarchy, since bottom-up, top-down, and the same-level associations are all done by similarity of representations. Under such an organization, generalization or similarity hierarchy for assemblies of parts and wholes is formed in different associative fields.

If generalization between items of different complexity is desirable, formation of their assemblies in the same associative field may be considered. In one version, part-whole relationships could be formed explicitly, using binding with the roles of parts and wholes, represented, e.g., by special codevectors (?see also section 8). In another version, present multi-module architecture could be augmented with a common module where the item assemblies of various compositional levels are accumulated for the purpose of generalization (?see also section ....). Related potential problems of how similar items from diverse conceptual and linguistic structures might be uniformly allocated to the proper levels of part-whole hierarchies and how the items that seem cross levels might be handled also need further investigation.

Other problems connected with the context and mentioned in section 10.2 are how the attributes are segmented, how their small groups that form chunks are selected, and how large groups of attributes characteristic to an item are formed from small chunks. Probably, the chunks of attributes are accumulated in memory forming a rather "branchy" (disperse) assembly corresponding to an item.

10.5. The is-a or taxonomy hierarchy

In traditional AI and in object-oriented design or programming, the structure of classes is designed by the system developer for each domain and problem. In an adaptive intelligent system, classification hierarchies must substantially emerge by learning. This requires further investigation of associative field functioning for real, correlated representations, including formation of category-based hierarchy in neural assemblies, storage capacity of memory, mechanisms of retrieval and traveling through the hierarchy.

The usual version of the Hebbian learning rule permits an easy activation of the core part of an assembly, corresponding to categories or prototypes, but the assembly parts corresponding to a category instance are difficult to activate. It is necessary to investigate further the influence of variations in the learning rules for correlated items on prototype forming and on the possibility to activate separately neural assemblies that differ from each other only by a small fraction of neurons. It is known, for example, that during the learning process humans often emphasize the differences of items, and this can be taken into account in the learning rules.

Various mechanisms for activity control and their influence on the functioning of the associative field and on the abstraction level of concept also need further investigation. In particular, these mechanisms could include both forced external changing of activity level in the network by

threshold control and exciting assembly parts with certain characteristics of absolute or relative connectivity. For all mechanisms of activity control and for various learning rules, it is necessary to examine the possibility of stable activation of assemblies with complex internal structure corresponding to category prototypes and instances, investigate storage capacity, and study the context influence.

10.6. Interaction of part-whole and category-based hierarchies

Formation of a category-based assembly structure leads to a non-uniform connectivity inside assemblies. During decoding, this complicates retrieval of component assemblies provided with a composite assembly, demands taking into account and controlling the level of generality.

Context-Dependent Thinning is a key procedure in part-whole hierarchy formation, however it makes the structure of assemblies more complex and complicates the analysis of category-based hierarchy formation. It is necessary to investigate the influence of thinning on the generalization ability of assembly networks.

10.7. New learning rules

Unlike the learning rules for multi-layer perceptrons, that demand outside storage and repeated presentation of the training set, training in APNNs forms internal representation of the learned items. Then they can be compared with the perceived items, and the information about their similarities and differences can be used to facilitate learning. Not only averaged prototypes are formed in the network, but instance items may be stored as well in parallel with the prototypes. At the same time, in the process of learning, hierarchies of concepts of various degree of generality are naturally formed.

10.8. Emotions, feelings, estimations

We did not consider emotions, feelings, estimations, etc. in detail in this article, however they have great importance for autonomous operation of intelligent systems. This matter receives a fair amount of attention in the literature, and we share many proposals, e.g., of Hebb (1949) and Amosov (1967), concerning these issues.

10.9. Prospective applications

As for other schemes of structure-sensitive distributed representations (e.g., Plate 1999), the following areas seem promising for APNN applications:
- analogical reasoning and other kinds of reasoning and inferencing. A substantial information about human analogical reasoning has been accumulated by psychologists (e.g., Gentner & Markman 1995; Hummel & Holyoak 1997; Keane 1997), and efforts could be undertaken to take it into account by the APNN-based models.
- advanced information (document) access, where the techniques of unstructured vector space models (Deerwester et.al. 1990; Caid, Dumais, & Gallant 1995; Kanerva et.al. 2000) could be augmented with the structural information extracted from the texts.
- natural language processing is a closely related area, where merging APNNs with the approaches of Steedman (1999); Seidenberg & MacDonald (1999); Hadley et.al. (2000) looks promising (see also Christiansen, Chater, & Seidenberg 1999; Nirenburg.& Raskin.2001).
- visual and acoustical recognition using the information about structure of items (Marr 1982; Hummel & Beiderman1992; Hummel 2000; but see also Intrator & Edelman 2000, Edelman 1998).

The list of problems and directions of future research above does not pretend to be full. We hope the BBS commentary will extend it and will impart impulse to the research in the field of structured distributed representations.

11. Conclusions

In this article, we have presented APNNs as an example of the neural network architecture that uses distributed representations for building a rich, complex model of the world. The world model is constructed from various item models organized in part-whole and category-based types of hierarchy.

The APNN representations allow a large number of models of various complexity to be encoded in codevectors of the same dimensionality. Representations of composite item models comprise reduced representations of their component models or attributes, such as global properties and parts with their relationships and roles. These component representations may also consist from reduced representations of their own components. Therefore, the content of a composite item codevector is influenced by representations of its component models and their relations of all previous levels of the part-whole hierarchy.

Similarity of reduced representations of composite models depends on the similarity of their components, on their grouping and order, and can be estimated immediately, without decoding and mapping of their structure. Besides, similarity of representations serves as the basis for various kinds of associations between the models.

Further work should reveal how potentially attractive from the scaling point of view schemes for distributed structure representation, processing algorithms exploiting them, and implementations of long-term memory will actually behave themselves for very large numbers of structured items.

We hope that under an appropriate elaboration, the concepts and techniques described in this article will provide a unified basis for creating well-scaling models of various aspects of intelligent behavior, from pattern recognition, action planning, and motoric control to language processing, analogical reasoning, and decision making. The uniform approach to encoding and processing of information of different hierarchical levels and modalities will also make it possible to unite developed fragments in the framework of single neural network structure. It may also show the ways to implementation of existing cognitive models, such as "perceptual symbols" of Barsalou (1999).

The APNN scheme might have some degree of neurological plausibility and provide a source of analogies between descriptions of brain functions at psychological and neurophysiological levels, e.g., between human memory capacity and storage capacity of neural network, variability of human concept structure and diversity of context-dependent activations of a neural assembly, concept generalization/differentiation in the process of human learning and assembly formation during neural network learning, etc.

We believe that APNN-like architectures based on structured compositional distributed representations and distributed memory will be useful for AI applications capable of handling complex real-world data and solving a wide range of structure-sensitive AI problems in various domains, and combining them in application systems (such as autonomous robots). This is facilitated by combining manipulation of symbols (basis of traditional AI) and pattern matching (basis of traditional neural networks) in a single framework. Development of computers also promotes this approach, making it already possible to implement such models on PCs.

References

Amari, S. (1989). Characteristics of sparsely encoded associative memory. Neural Networks, 2, 445-457.

Amit, D.J. (1989) Modeling brain function: The world of attractor neural networks Cambridge University Press

Amit, D.J. (1995) THE HEBBIAN PARADIGM REINTEGRATED: LOCAL REVERBERATIONS AS INTERNAL REPRESENTATIONS - Behavioral and Brain Science Dec. 1995 - Daniel J. Amit

Amosov, N. M. (1967). Modelling of thinking and the mind. New York: Spartan Books.

Amosov, N. M. (1979). Algorithms of the Mind. Kiev: Naukova Dumka (in Russian)

Amosov, N. M., Baidyk, T. N., Goltsev, A. D., Kasatkin, A. M., Kasatkina, L. M., Kussul, E. M., & Rachkovskij, D. A. (1991). Neurocomputers and intelligent robots. Kiev: Naukova dumka. (In Russian).

Amosov, N.M., Kasatkin, A.M., & Kasatkina L.M.(1975) Active semantic networks in robots with an autonomous control. Advance papers of the Fourth Intern. Joint Conference on Artificial intelligence v.9 pp. 11-20 (in Russian. English version exists but not available to me).

Amosov, N.M., Kasatkin, A.M., Kasatkina, L.M., Talayev, S.A. (1973). Automata and the mindfull behaviour. Kiev: Naukova Dumka (in Russian).

Amosov, N.M., & Kussul, E.M. Possible structure of system for reinforcement and inhibition. In: "Problems of heuristic modelling", Inst. of Cybernetics, Ukrainian Acad. Sci., 1969, no.1, pp. 3-11 (in Russian).

Amosov, N.M., Kussul, E.M., & Fomenko, V.D. (1975) Transport robot with a neural network control system. Advance papers of the Fourth Intern. Joint Conference on Artificial intelligence v.9 p.1-10 (In Russian. English version exists but not available to me).

Anderson J.A. (1972) A simple neural network generating an interactive memory. Mathematical Biosciences, 1970, 197-220

Anderson J.A. (1983). Cognitive and Psychological Computation with Neural Models. IEEE transactions on Systems, Man, and Cybernetics, SMC-13, No 5, 799-814.

Anderson, J.A. & Hinton, G.E., eds. (1981) Parallel models of associative memory. Hillside, NJ: Lawrence Erlbaum Associates.

Anderson J.A., Murphy G.L. (1986). Psychological Concepts in a Parallel System. Physica 22D, 318-336.

Anderson, J.A., Silverstein, J.W., Ritz, S.A., & Jones, R.S. Distinctive features, categorical perception, and probability learning: Some application of a neural model. Psychological Review, 1977, 84, 413-451.

Anderson, J.R., & Bower, G.H. (1973). Human associative memory. Washington, DC: V.H.Winston.

Antomonov, Yu.G. (1969). Systems, Complexity, Dynamics. Kiev: Naukova Dumka (in Russian)

Antomonov, Yu.G. (1974). Principles of Neurodynamics. Kiev: Naukova Dumka (in Russian)

Artykutsa, S. Ya., Baidyk, T. N., Kussul, E. M., & Rachkovskij, D. A. (1991). Texture recognition using neurocomputer. (Preprint 91- 8). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Ashby, W.R. (1956) An Introduction to Cybernetics. London: Chapman & Hall.

Baidyk T.N., Kussul E.M. (1992) Structure of neural assembly. In Proceedings of The RNNS/IEEE Symposium on Neuroinformatics and Neurocomputers. Rostov-on-Don, Russia. pp. 423-434

Baidyk, T. N., Kussul, E. M., & Rachkovskij, D. A. (1990). Numerical-analytical method for neural network investigation. In Proceedings of The International Symposium on Neural Networks and Neural Computing - NEURONET'90 (pp. 217-219). Prague, Czechoslovakia.

Barsalou, L.W. (1999). Perceptual symbol systems. Behavioral and Brain Sciences, 22, 577-609.

Baum, E.B., J. Moody, and F. Wilczek (1988). Internal representations for associative memory. Biological Cybernetics 59, 217-228.

Booch, G (1994) Object-oriented analysis and design. Benjamin/Cummings. Menlo Park, Calif.

Braitenberg V. Cell assemblies in the cerebral cortex. In: Theoretical approaches to complex systems, 171-188, Heim R. and Palm G. (eds.), Springer Verlag, New York, 1978. Braitenberg V. Cell assemblies in the cerebral cortex. Lect. Notes Biomath. - 1978. - 21. - P.171-188.

Caid, WR, Dumais ST, & Gallant SI. Learned vector-space models for document retrieval. Information Processing and Management, Vol. 31, No. 3, pp. 419-429, 1995.

Cangelosi, A. & Harnad, S. (2000) The Adaptive Advantage of Symbolic Theft Over Sensorimotor Toil:Grounding Language in Perceptual Categories. Evolution of Communication (Special Issue on Grounding)

Carpenter, G.A., Grossberg, S. (1987) A massively parallel architecture for a selforganizing neural pattern recognition machine Computer Vision, Graphics and Image processing 37, pp. 54-115.

Christiansen, M.H., Chater, N. & Seidenberg, M.S. (Eds.) (1999). Connectionist models of human language processing: Progress and prospects. Special issue of Cognitive Science, Vol. 23(4), 415-634.

Cowan, N. (2001) The Magical Number 4 in Short-term Memory: A Reconsideration of Mental Storage Capacity. Behavioral and Brain Sciences 24 (1): XXX-XXX.

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, & R. A. Harshman. Indexing by Latent Semantic Analysis. Jornal of the American Society for Information Science, 41(6): 391-407, 1990.

Dorffner, G & E. Prem. Connectionism, Symbol Grounding, and Autonomous Agents. Technical Report TR-93-17, Austrian Research Institute for AI.

Edelman, S. (1998). Representation is representation of similarities. Behavioral and Brain Sciences, 21, 449-498.

Eliasmith, C.& Thagard, P. (in press) Integrating Structure and Meaning: A Distributed Model of Analogical Mapping. Cognitive Science.

Falkenhainer, B., Forbus, K.D., & Gentner, D. (1989). The Structure-Mapping Engine: Algorithm and Examples. Artificial Intelligence, 41, 1-63.

Fedoseyeva, T. V. (1992). The problem of training neural network to recognize word roots. In Neuron-like networks and neurocomputers (pp. 48-54). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Feigelman M.V., Ioffe, L.B. (1987) The augmented models of associative memory - asymmetric interaction and hierarchy of patterns. International Journal of Modern Physics, B1, 51-68.

Feldman, J. A. (1989). Neural Representation of Conceptual Knowledge. In L. Nadel, L. A. Cooper, P. Culicover, & R. M. Harnish (Eds.), Neural connections, mental computation (pp. 68- 103). Cambridge, Massachusetts, London, England: A Bradford Book, The MIT Press.

Forbus, K.D., Gentner, D., & Law, K. (1995). MAC/FAC: A model of similarity-based retrieval. Cognitive Science, 19(2), 141- 205.

Frasconi, P., Gori, M., & Sperduti, A. (1997). A general framework for adaptive processing of data structures. Technical Report DSI- RT-15/97. Firenze, Italy: Universita degli Studi di Firenze, Dipartimento di Sistemi e Informatica.

Frolov, A. A., & Muraviev, I. P. (1987). Neural models of associative memory. Moscow: Nauka. (In Russian).

Frolov, A. A., & Muraviev, I. P. (1988). Informational characteristics of neuronal and synaptic plasticity. Biophysics, 33, 708-715.

Frolov A.A., Husek D., Muraviev I.P. (1997) Informational capacity and recall quality in sparsely encoded Hopfield-like associative memory. Neural Networks. 9, pp. 1012-1025.

Gayler, R. W. (1998). Multiplicative binding, representation operators, and analogy. In K. Holyoak, D. Gentner, & B. Kokinov (Eds.), Advances in analogy research: Integration of theory and data from the cognitive, computational, and neural sciences. (p. 405). Sofia, Bulgaria: New Bulgarian University. (Poster abstract. Full poster available at: http://cogprints.soton.ac.uk/abs/comp/199807020).

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

Gentner, D. & Markman, A. B. (1995). Analogy-Based Reasoning. In M. A. Arbib (Ed.), Handbook of brain theory and neural networks (pp. 91-93). Cambridge, MA: MIT Press.

Gentner, D., & Markman, A. B. (1997). Structure Mapping in Analogy and Similarity. American Psychologist, 52(1), 45-56.

Gladun, V.P. (1977) Heuristic search in complex environments Kiev: Naukova Dumka. (in Russian)

Gladun V.P. (1994) Processes of new knowledge formation. Sofia: SD "Pedagog 6". (in Russian)

Goltsev A.D. Investigation of mechanisms for assembly organization in neural networks. Ph.D. Thesis, Kiev, 1976 (in Russian).

Goltsev, A. (1996). An assembly neural network for texture segmentation. Neural Networks, 9(4), 643-653.

Gray, B., Halford, G. S., Wilson, W. H., & Phillips, S. (1997, September 26-28). A Neural Net Model for Mapping Hierarchically Structured Analogs. Proceedings of the Fourth conference of the Australasian Cognitive Science Society, University of Newcastle.

Grossberg, S.Pavlovian pattern learning by nonlinear neural networks. Proceedings of the National Academy of Sciences, 1971, 68, 828-831.

Grossberg, S. (1982) Studies of Mind and Brain: Neural principles of learning, perception, development, cognition and motor control. Boston: Reidel.

Gutfreund H. (1988) Neural network with hierarchically correlated patterns. Physical Review A37, pp.570- 577.

Hadley, R.F., Rotaru-Varga, A., Arnold, D.V., and Cardei, V.C. (2000) ``Syntactic Systematicity Arising from Semantic Predictions in a Hebbian-Competitive Network", Submitted for Journal Review.

Harnad, S. (1987). Category induction and representation. In S. Harnad (Ed.), Categorical perception: The groundwork of cognition (pp. 535-565). New York: Cambridge University Press.

Harnad, S. (1990). The Symbol Grounding Problem. Physica D 42:335-346.

Hebb, D. O. (1949). The organization of behavior. New York: Wiley.

Hinton, G. E. (1990). Mapping part-whole hierarchies into connectionist networks. Artificial Intelligence, 46, 47-76.

Hinton, G. E., McClelland, J. L., & Rumelhart, D. E. (1986). Distributed representations. In D. E. Rumelhart, J. L. McClelland, & the PDP research group (Eds.), Parallel distributed processing: Exploration in the microstructure of cognition 1: Foundations (pp. 77-109). Cambridge, MA: MIT Press.

Hirahara, M., O. Oka, T. Kindo (2000) Cascade associative memory storing hierarchically correlated patterns with various correlations Nneurl Networks 13(1), pp.51-61.

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

Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences, USA, 79, 2554-2558.

Hopfield, J. J., Feinstein, D. I., & Palmer, R. G. (1983). "Unlearning" has a stabilizing effect in collective memories. Nature, 304, 158- 159.

Hummel, J. E. (2000). Where view-based theories break down: The role of structure in shape perception and object recognition. In E. Dietrich and A. Markman (Eds.). *Cognitive Dynamics: Conceptual Change in Humans and Machines* (pp. 157 - 185). Hillsdale, NJ: Erlbaum.

Hummel, J.E., & Beiderman, I. (1992) Dynamic binding in a neural network for shape recognition. Psychological Review, 99, 480-517.

Hummel, J. E., & Holyoak K. J. (1997). Distributed representations of structure: A theory of analogical access and mapping. Psychological Review, 104, 427-466.

Intrator, N & Edelman, S. (2000) REPRESENTING THE STRUCTURE OF VISUAL OBJECTS. Titles, abstracts, and background material provided by the speakers of NIPS'2000 workshop. http://kybele.psych.cornell.edu/~edelman/NIPS00/material.html

Kakeya, H., & Y.Okabe (1999) Selective retrieval of Memory and concept sequences through neuro-windows. IEEE Transactions on Neural Networks, v.10, No.1, pp. 182-185.

Kanerva, P. (1988). Sparse distributed memory. Cambridge, MA: MIT Press.

Kanerva, P. (1994). The Spatter Code for encoding concepts at many levels. In M. Marinaro and P.G. Morasso (eds.), ICANN '94, Proceedings of International Conference on Artificial Neural Networks (Sorrento, Italy), 1, pp. 226-229. London: Springer- Verlag.

Kanerva, P. (1996). Binary Spatter-Coding of Ordered K-tuples. In C. von der Malsburg, W. von Seelen, J. C. Vorbruggen, & B. Sendhoff (Eds.). Proceedings of the International Conference on Artificial Neural Networks - ICANN'96, Bochum, Germany. Lecture Notes in Computer Science, 1112, 869-873. Berlin: Springer.

Kanerva, P. (1998). Encoding structure in Boolean space. In L. Niklasson, M. Boden, and T. Ziemke (eds.), ICANN 98: Perspectives in Neural Computing (Proceedings of the 8th International Conference on Artificial Neural Networks, Skoevde, Sweden), 1, pp. 387-392. London: Springer.

Kanerva, P., Kristoferson, J., and Holst, A. (2000) "Random indexing of text samples for Latent Semantic Analysis." In L.R. Gleitman and A.K. Josh (eds.), Proc. 22nd Annual Conference of the Cognitive Science Society (U Pennsylvania), p. 1036. Mahwah, New Jersey: Erlbaum.

Kasatkin, A. M., & Kasatkina, L. M. (1991). A neural network expert system. In Neuron-like networks and neurocomputers (pp. 18-24). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Keane, M.T. (1997) What makes an analogy difficult? The effects of order and causal structure on analogical mapping. Journal of Experimental Psychology: Learning, Memory, and Cognition, 23, no 4, 946-967.

Kohonen T (1972) Correlation matrix memories. IEEE Transactions on Computers, C-21, 353-359.

Kussul, E. M. (1980). Tools and techniques for development of neuron-like networks for robot control. Unpublished Dr. Sci. dissertation. Kiev, Ukrainian SSR: V. M. Glushkov Institute of Cybernetics. (In Russian).

Kussul, E. M. (1988). Elements of stochastic neuron-like network theory. In Internal Report "Kareta-UN" (pp. 10-95). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Kussul, E. M. (1992) Associative neuron-like structures. Kiev: Naukova Dumka. (In Russian).

Kussul, E. M. (1993). On some results and prospects of development of associative-projective neurocomputers. In Neuron-like networks and neurocomputers (pp. 4-11). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Kussul, E. M., & Baidyk, T. N. (1990). Design of a neural-like network architecture for recognition of object shapes in images. Soviet Journal of Automation and Information Sciences, 23(5), 53-58.

Kussul, E. M., & Baidyk, T. N. (1993a). On information encoding in associative-projective neural networks. (Preprint 93-3). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Kussul, E. M., & Baidyk, T. N. (1993b). A modular structure of associative-projective neural networks. (Preprint 93-6). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Kussul, E. M., Baidyk, T. N., Lukovich, V. V., & Rachkovskij, D. A. (1993). Adaptive neural network classifier with multifloat input coding. In Proceedings of NeuroNimes'93, Nimes, France, Oct. 25- 29, 1993. EC2-publishing.

Kussul, E.M., T.N. Baidyk, VV. Lukovich, D.A. Rachkovskij, Adaptive High Performance Classifier Based on Random Threshold Neurons, Proc. of Twelfth European Meeting on Cybernetics and Systems Research (EMCSR- 94), Austria, Vienna, 1994, pp. 1687-1695.

Kussul E.M.,& Fedoseyeva T.V. On audio signls recognition in neural assembly structures. Kiev, 1987, 21 pp, Prepr.87-28, Inst. Of Cybernetics, Ukrainian Acad. Sci. (in Russian).

Kussul, E. M., & Kasatkina, L. M. (1999). Neural network system for continuous handwritten words recognition. In Proceedings of the International Joint Conference on Neural Networks. (Washington, DC).

Kussul, E.M., Lukovich, V.V, Lutsenko, V.N. (1988) Multiprocessor computational devices for robot control in natural environment. Control systems and machines, no 5, pp.102-105 (in Russian).

Kussul, E. M., & Rachkovskij, D. A. (1991). Multilevel assembly neural architecture and processing of sequences. In A .V. Holden & V. I. Kryukov (Eds.), Neurocomputers and Attention: Vol. II. Connectionism and neurocomputers (pp. 577- 590). Manchester and New York: Manchester University Press.

Kussul, E. M., Rachkovskij, D. A., & Baidyk, T. N. (1991a). Associative-Projective Neural Networks: architecture, implementation, applications. In Proceedings of the Fourth International Conference "Neural Networks & their Applications", Nimes, France, Nov. 4-8, 1991 (pp. 463-476).

Kussul, E. M., Rachkovskij, D. A., & Baidyk, T. N. (1991b). On image texture recognition by associative-projective neurocomputer. In C. H. Dagli, S. Kumara, & Y. C. Shin (Eds.), Proceedings of the ANNIE'91 conference "Intelligent engineering systems through artificial neural networks" (pp. 453-458). ASME Press.

Kussul, E.M., D.A. Rachkovskij, and D. Wunsch (1999) The Random Subspace coarse coding scheme for real-valued vectors In: Proceedings of the International Joint Conference on Neural Networks, Washington, DC, July 10-16, 1999.

Lansner, A., & Ekeberg, O. (1985). Reliability and speed of recall in an associative network. IEEE Trans. Pattern Analysis and Machine Intelligence, 7, 490-498.

Lavrenyuk, A. N. (1995). Application of neural networks for recognition of handwriting in drawings. In Neurocomputing: issues of theory and practice (pp. 24-31). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Legendy, C. R. (1970). The brain and its information trapping device. In Rose J. (Ed.), Progress in cybernetics, vol. I. New York: Gordon and Breach.

Marr, D. (1969). A theory of cerebellar cortex. Journal of Physiology, 202, 437-470.

Marr, D. (1982). *Visio*n. Freeman: New York.

Markman, A.B. (1997). Structural alignment in similarity and its influence on category structure. Cognitive Studies, 4(4), 19-37.

Marshall, J.A. (1990) A self-organizing scale-sensitive neural network. Proceedings of the International Joint Conference on Neural Networks, San Diego, CA, vol. 3, 649-654.

McClelland, J.L., Rumelhart, D.E. (1986) A distributed model of human learning and memory. pp. 171-215. in McClelland, J.L., Rumelhart, D.E., & the PDP Research Group (1986). Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 2. Psychological and biological models. Cambridge, MA: MIT Press.

Milner P.M. The cell assembly: Mark II. Psychol. Rev., 64, 242-252 (1957).

Milner, P. M. (1974). A model for visual shape recognition. Psychological Review, 81, 521-535.

Milner, P.M. (1996). Neural representations: some old problems revisited. Journal of Cognitive Neuroscience, 8, 69-77.

Minsky, M. (1975). A framework for representing knowledge. In P.H. Winston (Ed.), The psychology of computer vision (pp. 211-277)!. New York: McGraw-Hill.

Nadal, J.-P, & G Toulouse (1990) Information storage in sparsely coded memory nets Network: Comput. Neural Syst. 1 No 1 61-74

Nakano K. (1972) Associatron - A model of associative memory. IEEE transactions on Systems, Man, and Cybernetics. Vol.SMC2(3), pp.380-388.

Nirenburg, S. and V. Raskin.(2001) Ontological Semantics.

Page, M. Connectionist Modelling in Psychology: A Localist Manifesto Behavioral and Brain Sciences

Palm, G. (1980). On associative memory. Biological Cybernetics, 36, 19-31.

Palm, G. (1982) Neural Assemblies: An Alternative Approach to Artificial Intelligence. Springer Verlag, Berlin, 1982.

Palm, G. (1993). The PAN system and the WINA project. In P. Spies (Ed.), Euro-Arch'93 (pp. 142-156). Springer-Verlag

Parga, N. & Virasoro, M.A. (1986). The ultrametric organization of memories in a neural network. J. Physique, 47, 1857-1864.

Plate, T. A. (1991). Holographic Reduced Representations: Convolution algebra for compositional distributed representations. In J. Mylopoulos & R. Reiter (Eds.), Proceedings of the 12th International Joint Conference on Artificial Intelligence (pp. 30- 35). San Mateo, CA: Morgan Kaufmann.

Plate, T. A. (1995). Holographic Reduced Representations. IEEE Transactions on Neural Networks, 6, 623-641.

Plate, T. (1997). A common framework for distributed representation schemes for compositional structure. In F. Maire, R. Hayward, & J. Diederich (Eds.), Connectionist systems for knowledge representation and deduction (pp. 15-34). Queensland University of Technology.

Plate, T. (1999). Representation, Reasoning and Learning with Distributed Representations. Neural Computing Surveys 2, pp. 15-17 http://www.icsi.berkeley.edu/~jagota/NCS,

Plate, T.A. (2000). Structured Operations with Vector Representations. Expert Systems: The International Journal of Knowledge Engineering and Neural Networks, 17, 29-40.

Pollack, J. B. (1990). Recursive distributed representations. Artificial Intelligence, 46, 77-105.

Pulvermuller, F. (1999) Words in the brain's language Behavioral and Brain Sciences 22, 253-336.

Quillian, M.R. (1968) Semantic memory. In M.Minsky (Ed.), Semantic information processing, Cambridge, Mass.: MIT Press

Rachkovskij, D. A. (1990a). On numerical-analytical investigation of neural network characteristics. In Neuron-like networks and neurocomputers (pp. 13-23). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Rachkovskij, D. A. (1990b). Development and investigation of multilevel assembly neural networks. Unpublished Ph.D. dissertation. Kiev, Ukrainian SSR: V. M. Glushkov Institute of Cybernetics. (In Russian).

Rachkovskij, D. A. (1996). Application of stochastic assembly neural networks in the problem of interesting text selection. In Neural network systems for information processing (pp. 52-64). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Rachkovskij, D.A. (2001). Representation and processing of structures with binary sparse distributed codes. IEEE transactions on Knowledge and Data Engineering (Special Issue).

Rachkovskij, D. A. & Fedoseyeva, T. V. (1990). On audio signals recognition by multilevel neural network. In Proceedings of The International Symposium on Neural Networks and Neural Computing - NEURONET'90 (pp. 281-283). Prague, Czechoslovakia.

Rachkovskij, D. A. & Fedoseyeva T. V. (1991). Hardware and software neurocomputer system for recognition of acoustical signals. In Neuron-like networks and neurocomputers (pp. 62-68). Kiev, Ukraine: V. M. Glushkov Institute of Cybernetics. (In Russian).

Rachkovskij, D. A. & Kussul, E. M. (2001). Binding and Normalization of Binary Sparse Distributed Representations by Context-Dependent Thinning (paper draft available at http://cogprints.soton.ac.uk/abs/comp/199904008).

Rumelhart, D.E., Lindsay, P.H., & Norman, D.A. A process model for long-term memory. In E.Tulving & W. Donaldson (Eds.), Organization and memory. New York: Academic Press, 1972.

Rumelhart, D.E., Smolemsky, P., McClelland, J.L., & Hinton, G.E. (1986) Schemata and Sequential Thought Process in PDP Models. pp. McClelland, J.L., Rumelhart, D.E., & the PDP Research Group (1986). Parallel distributed processing: Explorations in the microstructure of cognition: Vol. 2. Psychological and biological models. Cambridge, MA: MIT Press.

Schank, R.C., & Abelson, R.P. (1977). Scripts, plans, goals, and understanding: An inquiry into human knowledge structures. Hillsdale, NJ: Lawrence Erlbaum Associates.

Seidenberg, M.S., & MacDonald, M.C. (1999). A probabilistic constraints approach to language acquisition and processing.Cognitive Science. 23, 569-588.

Shastri, L. & Ajjanagadde, V. (1993). From simple associations to systematic reasoning: connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. Behavioral and Brain Sciences, 16, 417-494.

Sjodin, G. (1998). The Sparchunk Code: a method to build higher- level structures in a sparsely encoded SDM. In Proceedings of IJCNN'98 (pp.1410-1415), IEEE, Piscataway, NJ: IEEE.

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

Sommer, F.T., & G. Palm (1999) Improved Bidirectional Retrieval of Sparse Patterns Stored by Hebbian Learning Neural Networks 12 (2) 281 - 297

Sperduti, A. (1994). Labeling RAAM. Connection Science, 6, 429- 459.

Steedman, M (1999) Connectionist Sentence Processing in Perspective, Cognitive Science, 23, 615-634.

Thagard, P., K.J.Holyoak, G.Nelson, D.Gochfeld Analog Retrieval by Constraint Satisfaction Artificial Intelligence 46(1990) 259-310

Thorpe, S. (1995) Localized versus distributed representations. In M.A.Arbib (Ed.), The Handbook of Brain Theory and Neural Networks, pp. 549-552. Cambridge, MA: MIT Press.

Tsodyks, M. V. (1989). Associative memory in neural networks with the Hebbian learning rule. Modern Physics Letters B, 3, 555-560.

Tsodyks M. V. (1990). Hierarchical associative memory in neural networks with low activity level. Modern Physics Letters B, 4, 259-265.

Vedenov, A. A., Ezhov A.A., L.A. Knizhnikova, E.B. Levchenko (1987). "Spurious memory" in model neural networks. (Preprint IAE-4395/1). Moscow: I. V. Kurchatov Institute of Atomic Energy. (in Russian)

Vedenov, A. A. (1988). Modeling of thinking elements. Moscow: Nauka. (In Russian).

von der Malsburg, C. (1981). The correlation theory of brain function. (Internal Report 81-2). Gottingen, Germany: Max-Planck- Institute for Biophysical Chemistry, Department of Neurobiology. von der Malsburg, C. (1985). Nervous structures with dynamical links. Ber. Bunsenges. Phys. Chem., 89, 703-710.

von der Malsburg, C. (1986) Am I thinking assemblies? In G. Palm & A. Aertsen (Eds.), Proceedings of the 1984 Trieste Meeting on Brain Theory (pp. 161-176). Heidelberg: Springer-Verlag.

Willshaw, D. (1981). Holography, associative memory, and inductive generalization. In G. E. Hinton & J. A. Anderson (Eds.), Parallel models of associative memory (pp. 83-104). Hillside, NJ: Lawrence Erlbaum Associates.

Willshaw, D. J., Buneman, O. P., & Longuet-Higgins, H. C. (1969). Non-holographic associative memory. Nature, 222, 960-962.

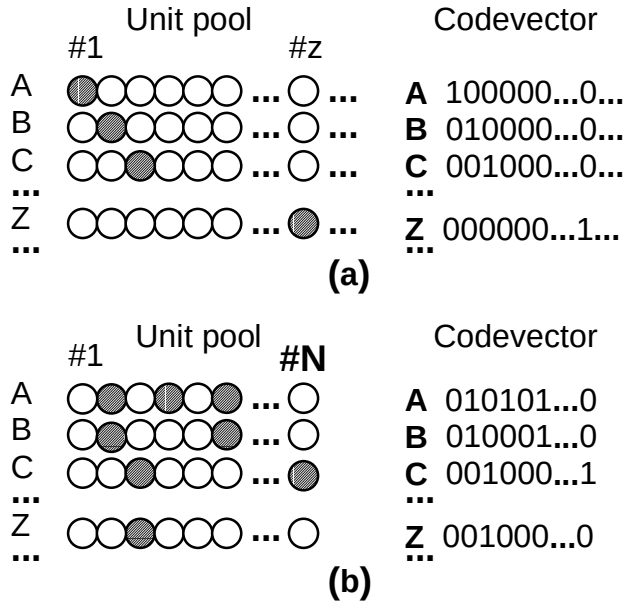

Figure 3.1. (a) Local and (b) distributed representations of elementary items.

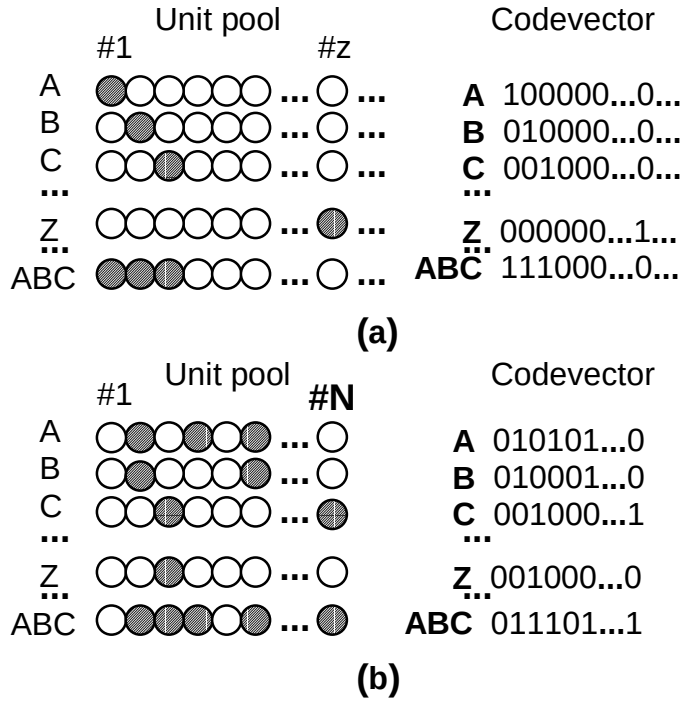

Figure 3.2. Fully-articulated or disjunctive representations of simple composite items. (a) Local and (b) distributed representations. ***ABC = AvBvC.***

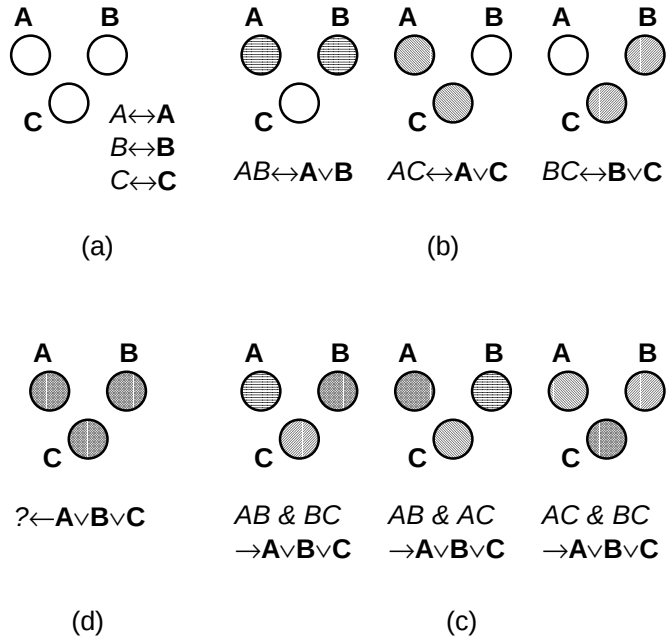

Figure 3.3. An illustration of "ghosts" or "superposition catastrophe". (a) Each component item *A*, *B*, *C* is represented by the "all-or-none" activity pattern **A**, **B**, **C** respectively. (b) Each composite item *AB*, *AC*, *BC* is represented by the superposition of activity patterns of the component items. (c) Superposition of activity patterns of any two of three composite items produces the representation in which the third unforeseen composite item (ghost) is encoded as well. (d) If all three component items are activated, it is impossible to tell which composite items are actually present (superposition catastrophe).

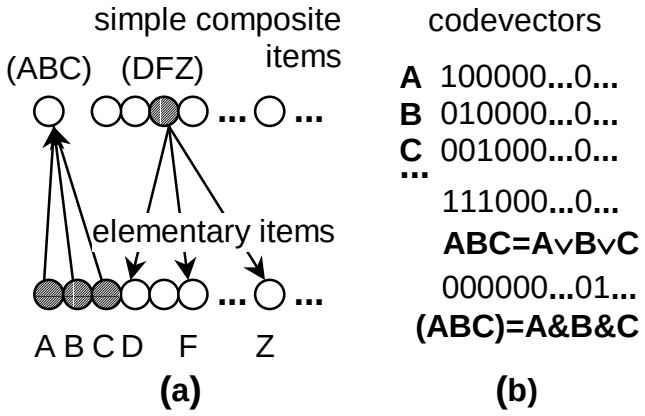

Figure 3.4. In local representations, a new unit (a) or a new element of the codevector (b) is introduced to represent simple composite items in the reduced form. This requires beating connections from the units of the components *A,B,C* to the unit *(ABC)*. Via these connections, activation of the component units leads to activation of the unit of their reduced representation. On the other hand, the bound or reduced representation may be decoded through their fully-articulated representations by following connections, e.g., from the *(DFZ)* unit to the *D,F,Z* units.

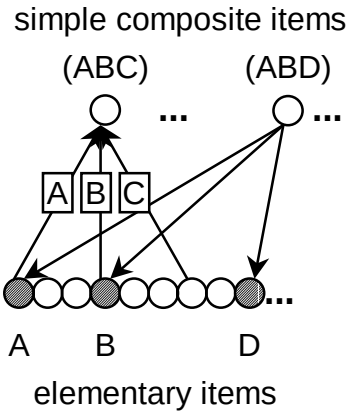

Figure 3.5. Each local reduced representation of a simple composite item makes up a "dot-product machine" for finding its similarity value to any other composite item activated in the fully-articulated fashion in the pool of elementary items. For example, the activation value of the *(ABC)* unit provides the value of its similarity to the items *A,B,D*. In order to find the similarity value of ***(ABC)*** to ***(ABD)***, the latter must be decoded through the elementary units. In order to find their similarity content, both must be decoded.

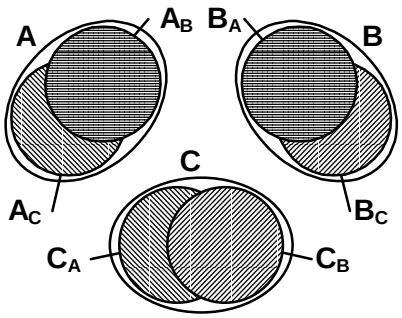

Figure 3.6 (first version). How thinning preserves similarity. Ovals represent the 1s encoding the component items *A*, *B*, *C*. Representations of three composite items *AB*, *AC*, *BC* formed by thinning are shown by the corresponding combinations of three small differently shaded circles. $\mathbf{X_Y}$ denotes the subset of 1s preserved in the thinned representation of item *X* when items *Y* is also present. It can be seen that $\mathbf{A_B}$, $\mathbf{A_C}$ are different subsets of **A.** Note, that actually the components are represented as randomly generated codevectors.

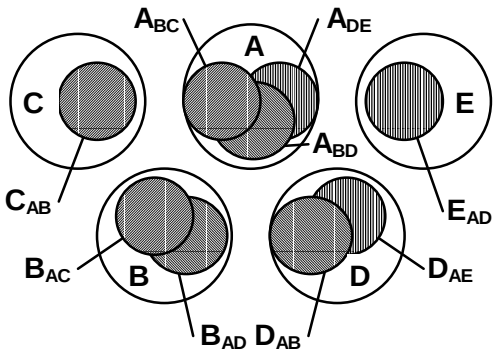

Figure 3.6 (second version) . How thinning preserves similarity. Big clear circles represent the 1s encoding the component items *A*, *B*, *C*, *D*, *E*. Representations of three composite items *ABD*, *ABC*, *ADE* formed by thinning are shown by the corresponding combinations of three small differently shaded circles. $\mathbf{X_{YZ}}$ denotes the subset of 1s preserved in the thinned representation of item *X* when items *Y* and *Z* are also present. It can be seen that $\mathbf{A_{BC}}$, $\mathbf{A_{BD}}$, and $\mathbf{A_{DE}}$ are all different subsets of **A**, and $\mathbf{A_{BC}}$ is more similar to $\mathbf{A_{BD}}$ than to $\mathbf{A_{DE}}$.

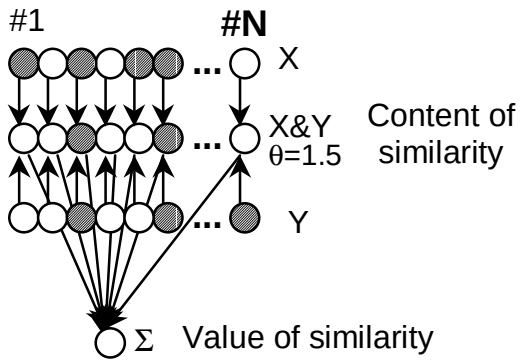

Figure 3.7. In distributed representations, a universal dot-product machine is used to calculate the similarity value and content of arbitrary codevectors ***X*** and ***Y*** activated over the unit pools. In local representations, a specialized dot-product machine exists for each item represented or memorized in reduced form (Figure 3.5).

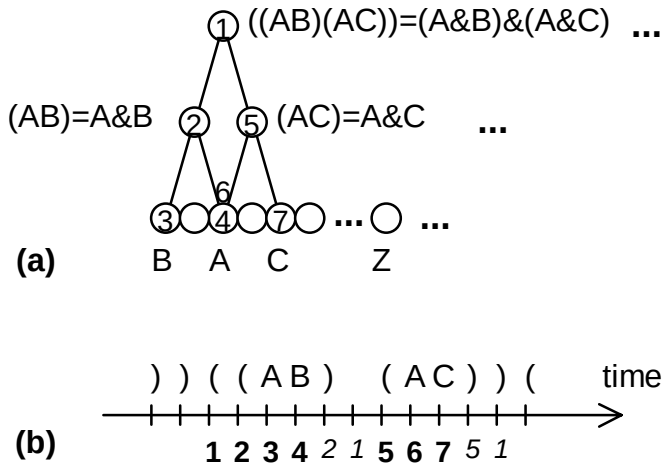

Figure 3.8. (a) Local representation of a hierarchical item by a growing tree or graph which can be implemented as a growing localist network. (b) Unfolding a hierarchical structure in time. Numbers show the sequence of decoding by following tree branches. Returns to the same unit are indicated in italic.

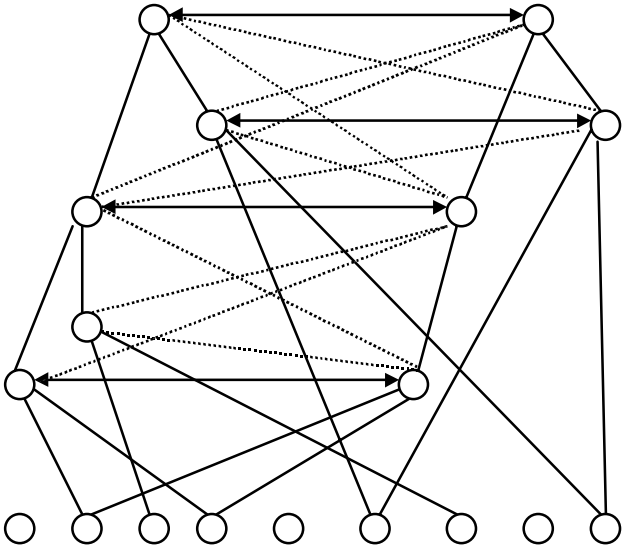

Figure 3.9. An estimation of the total similarity (semantic and structural similarity) of locally represented hierarchical structures requires computationally expensive alignment of their components. The arrows show probably the best match between the component nodes, and the dotted lines show some of the other possible candidate correspondences. The plain lines show “part-of” connections between the lower- and the higher-level units.

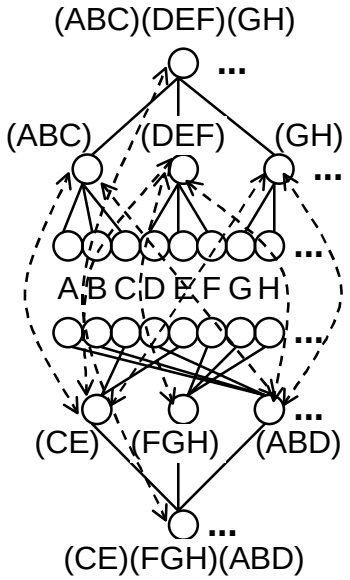

Figure 3.10. Finding similarities between representation trees of hierarchical items. Hatched (stroked) lines show possible correspondences between subitems.

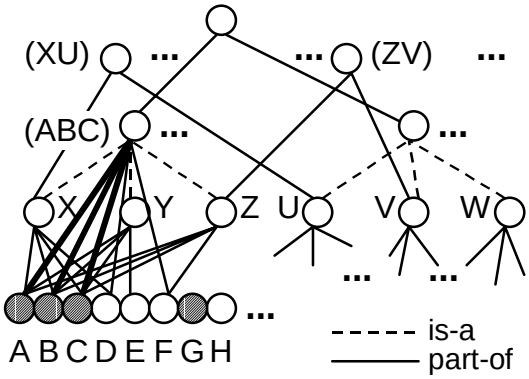

Figure 3.11. Local representations. Input *ABCG* activate composite item units *X,Y,Z*... Some other input elementary items activate *U,V,W,*... Which units should be used to construct items of the higher levels? If all activated units are used, we get a lot of higher-level units: (*XU)...(ZV)*...(see also Figure 3.12). Probably, class unit *(ABC)* should be used instead. It is connected to the specific units *X, Y, Z,*... containing component items *A, B, C*..., by *is-a* connections. It may also be directly connected to the component units *A,B,C,* by *part-of* connections of an increased strength.

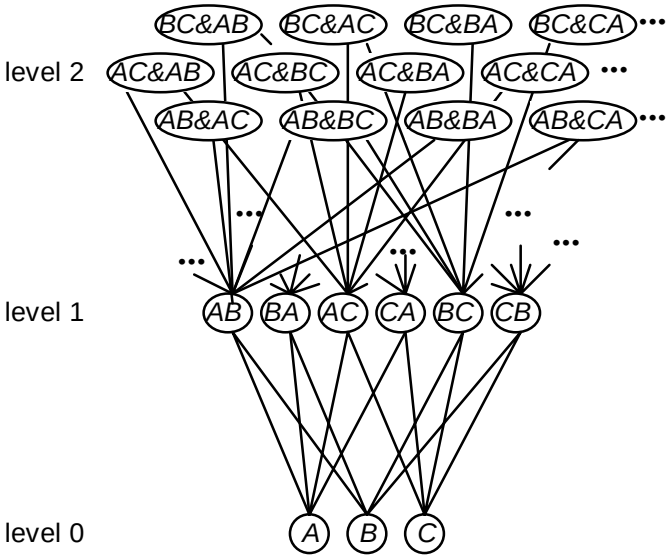

Figure 3.12 . Growth of the number of units which locally represent possible compositional structures versus the level of compositional hierarchy. (Here the order of items is taken into account).

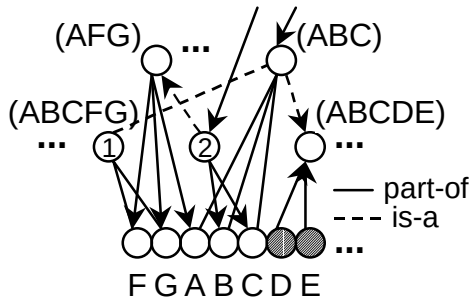

Figure 3.13. For local representations, the class *(ABC)* contains the components common to its instances *(ABCDE)*, *(ABCFG)*, etc. Then the instances must contain the extra, specific components - *D,E*, or *F,G*, etc. However, since any instance may belong to various classes (e.g., *(ABCFG)* belongs to*(ABC)*, *(AFG)*, etc.), it must be represented by various units - 1, 2, etc., each containing the complement, specific components for its class.
Components activated at the lower level may provide a context for activation of a particular instance of an activated class. E.g., once the class *(ABC)* activates one of its instances, *(ABCDE)*, provided with the activation of the components *D,E*.

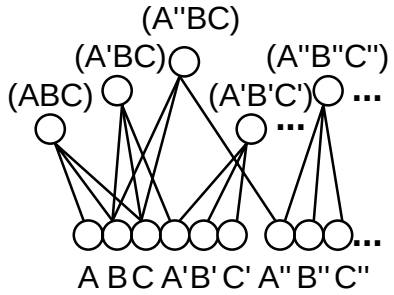

Figure 3.15. For local representations, each variant of a component item in various contexts requires a separate unit for the composite item.

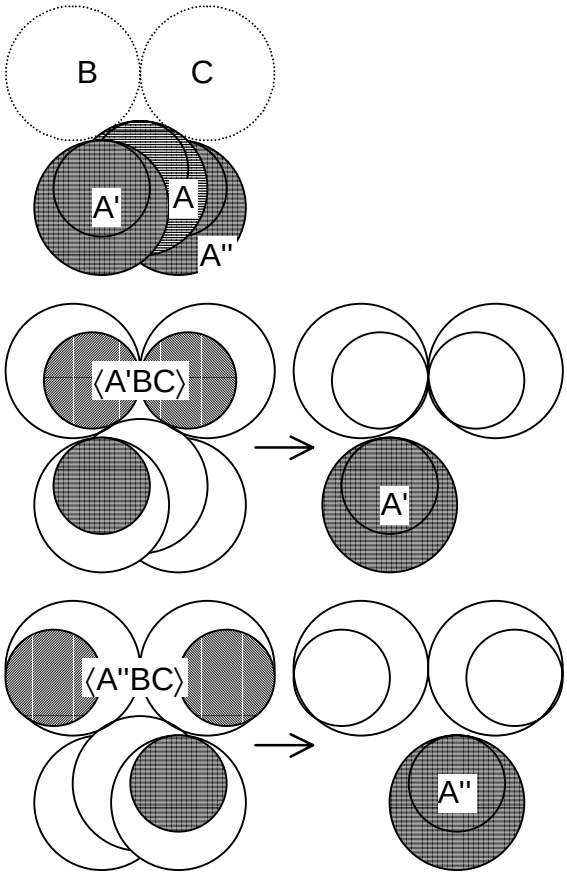

Figure 3.14. For distributed representations, both full and reduced representation of an item can be flexibly modified by the context. It influences the processes of retrieval and decoding. E.g., various full versions of reduced representations can be retrieved depending on the context.

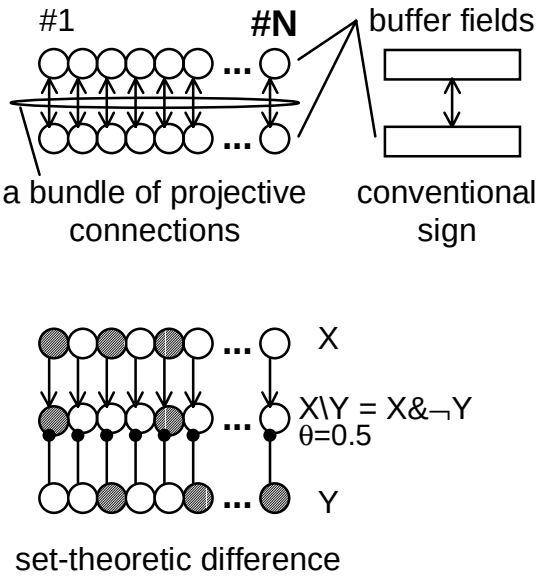

Figure 4.1. Buffer neural fields, connection bundles, and implementation of bitwise operations

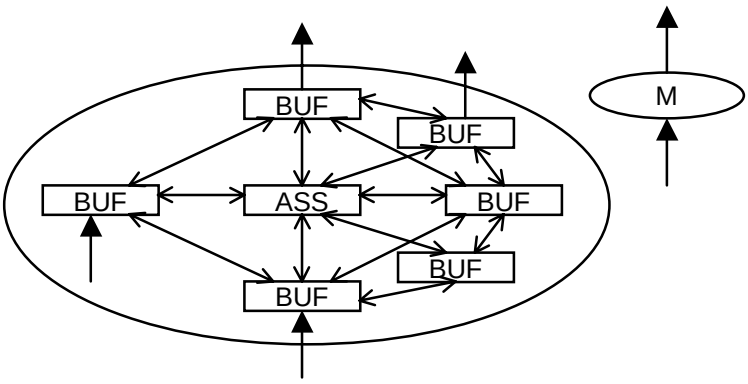

Figure 4.2. An example of the APNN module. Each module consists from a single associative field and a number of buffer fields. The fields are connected with each other by bundles of projective connections. The number of buffer fields and the specific structure of bundles between the fields depends on the specific problem. Inter-modular bundles of projective connections are shown with "filled" arrows. Filled end is on the side of the "higher-level" module. Intra-modular bundles of projective connections are shown by "open" bidirectional arrows.

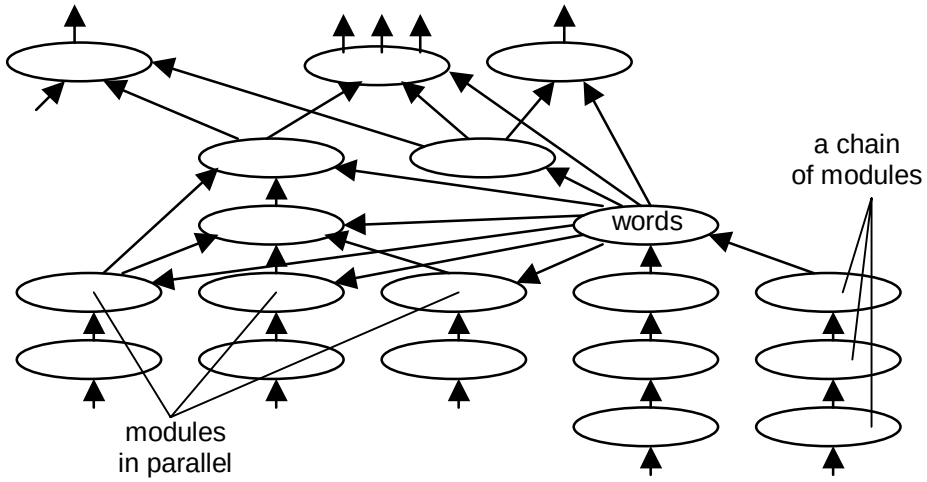

Figure 4.3. An example of the APNN architecture. The modules are connected by the bundles of projective connections shown by the "filled" arrows. The filled end is on the side of the "higher-level" module with more complex composite items.

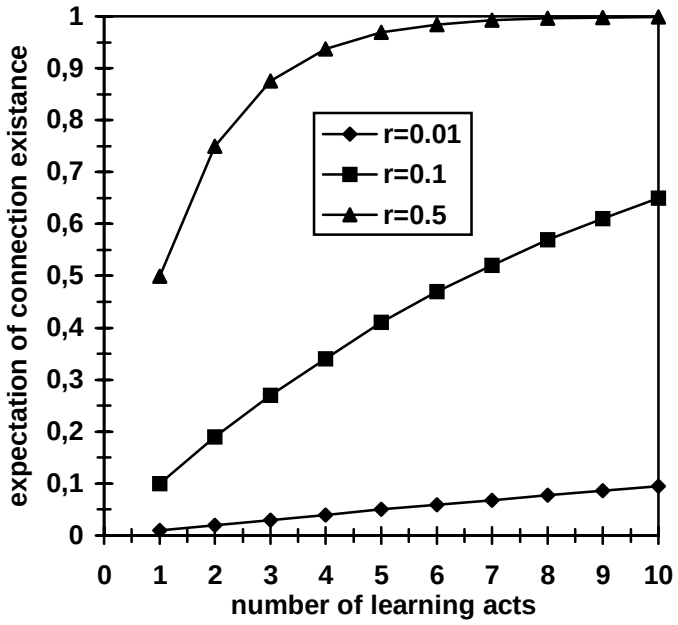

Figure 5.1. Probability of binary connections between two active neurons versus the number of learning acts.

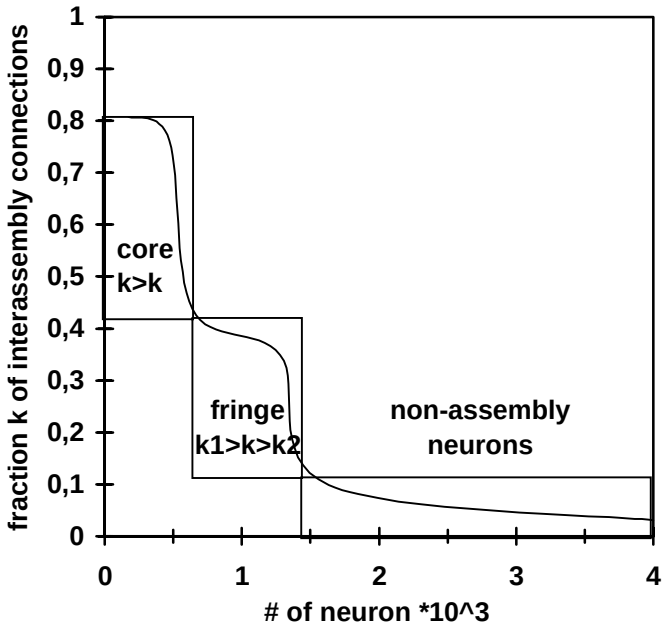

Figure 5.2. A schematic representation of an assembly structure. For some assembly, the neurons are ordered by the number of inter-assembly connections they have. Some non-assembly neurons are also shown.

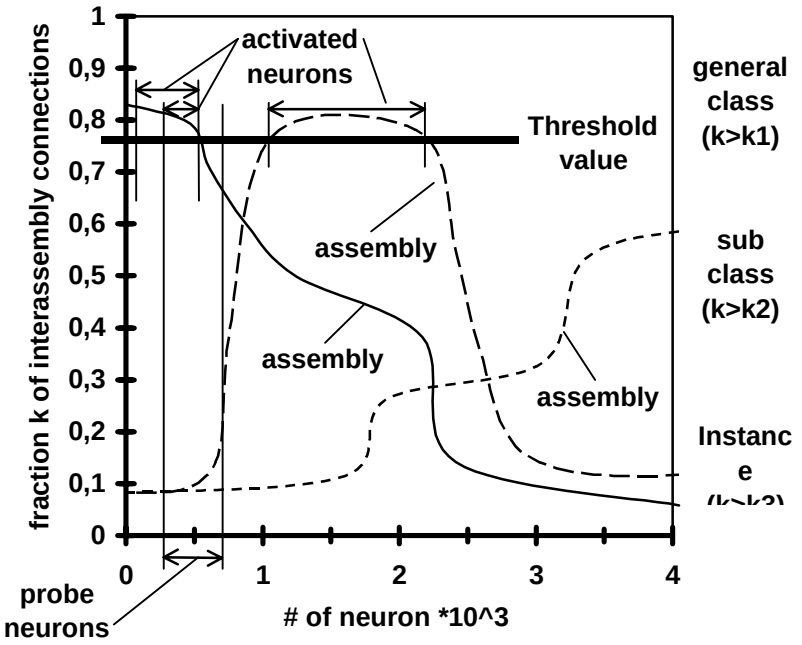

Figure 5.3. The activated assembly neurons are determined by the context of variously active neurons in the probe and inside the assemblies, the threshold value, as well as other factors, such as fatigue of neurons and connections. Depending on the particular context, the probe neurons (at the bottom) can activate very different parts of assemblies (at the top).

			$\mathbf{Z} \wedge \neg\tilde{\mathbf{Z}}(1) \wedge \neg\tilde{\mathbf{Z}}(2) \wedge \neg\tilde{\mathbf{Z}}(3) = \langle\mathbf{Z}\rangle$							
bit#	$\mathbf{X}_1$	$\mathbf{X}_2$	$\mathbf{Z}$	$\tilde{\mathbf{Z}}(1)$	$\neg\tilde{\mathbf{Z}}(1)$	$\tilde{\mathbf{Z}}(2)$	$\neg\tilde{\mathbf{Z}}(2)$	$\tilde{\mathbf{Z}}(3)$	$\neg\tilde{\mathbf{Z}}(3)$	$\langle\mathbf{Z}\rangle$
1	1	0	1	0	1	0	1	0	1	1
2	0	0	0	0	1	1	0	0	1	0
3	0	0	0	0	1	0	1	1	0	0
4	0	1	1	0	1	0	1	0	1	1
5	0	0	0	1	0	1	0	0	1	0
6	0	1	1	0	1	0	1	1	0	0
7	0	0	0	0	1	1	0	0	1	0
8	1	0	1	1	0	0	1	1	0	0
9	0	0	0	0	1	1	0	0	1	0
10	0	0	0	1	0	0	1	1	0	0
11	0	0	0	0	1	0	1	0	1	0
12	0	0	0	1	0	0	1	0	1	0

Figure 6.1 Toy example of the additive CDT procedure. 12-bit codevectors $\mathbf{X}_1$and $\mathbf{X}_2$ are first superimposed in $\mathbf{Z}$. Then $\mathbf{Z}$ is conjuncted with its shift permutations (4-bit shift, 1-bit shift, 2-bit shift down) until the number of 1s accumulated in resulting thinned codevector $\langle\mathbf{Z}\rangle$ reaches a predefined value. In our case, the number of 1s in $\langle\mathbf{Z}\rangle$ is equal to the number of 1s in each of the components $\mathbf{X}_1$ and $\mathbf{X}_2$.

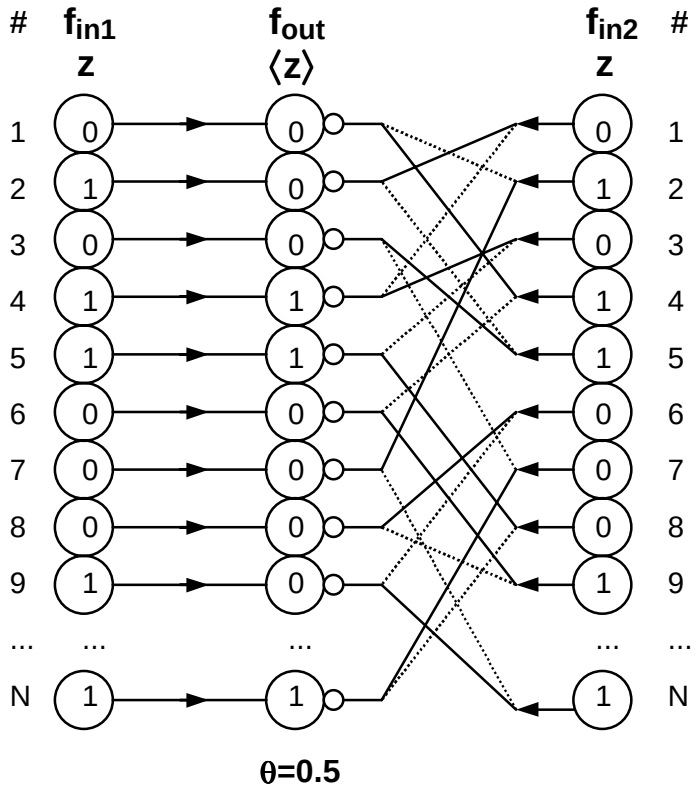

Figure 6.3. A neural-network implementation of the subtractive Context-Dependent Thinning procedure. There are three neuron fields of the same number of neurons: two input fields $\mathbf{f}_{in1}$ and $\mathbf{f}_{in2}$, as well as the output field $\mathbf{f}_{out}$. The copy of the input vector $\mathbf{z}$ is in both input fields. The neurons of $\mathbf{f}_{in1}$ and $\mathbf{f}_{out}$ are connected by the bundle of direct projective connections (1-to-1). The neurons of $\mathbf{f}_{in2}$ and $\mathbf{f}_{out}$ are connected by K bundles of independent permutive connections. (Only two bundles of permutive connections are shown here: one by solid lines, and one by dotted lines). The synapses of permutive connections are inhibitory (the weight is -1). The threshold of the output field neurons is 0.5. Therefore the neurons of $\mathbf{z}$ remaining active in $\mathbf{f}_{out}$ are those for which none of the permutive connections coming from $\mathbf{z}$ are active. As follows from Table 6.1, K is approximately the same for the number $S = 2,...,5$ of component codevectors of certain density p.

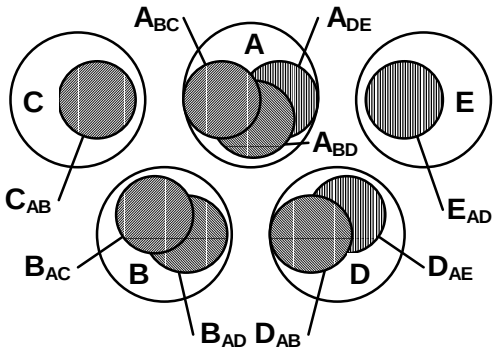

Figure 6.4. How thinning preserves similarity. Big clear circles represent the 1s encoding the component items *A*, *B*, *C*, *D*, *E*. Representations of three composite items *ABD*, *ABC*, *ADE* formed by thinning are shown by the corresponding combinations of three small differently shaded circles. $\mathbf{X_{YZ}}$ denotes the subset of 1s preserved in the thinned representation of item *X* when items *Y* and *Z* are also present. It can be seen that $\mathbf{A_{BC}}$, $\mathbf{A_{BD}}$, and $\mathbf{A_{DE}}$ are all different subsets of **A**, and $\mathbf{A_{BC}}$ is more similar to $\mathbf{A_{BD}}$ than to $\mathbf{A_{DE}}$.

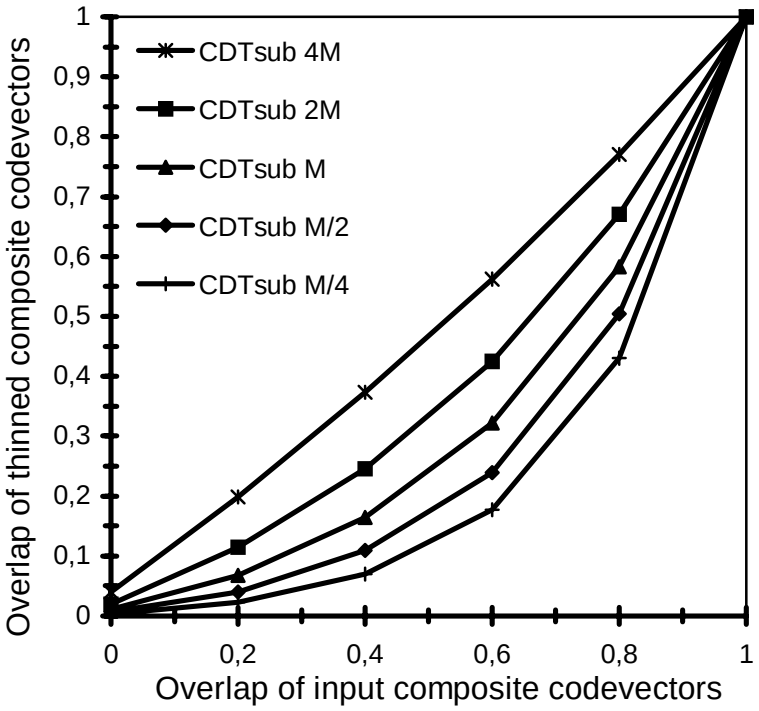

Figure 6.5. Overlap of thinned composite codevectors for various "depth" of thinning. There are five components in the composite item. For all component codevectors, N=100,000, $M\approx$1000. Therefore the input composite codevector includes about 5M of 1s. The resulting codevector was thinned to have from 4M to M/4 of 1s.

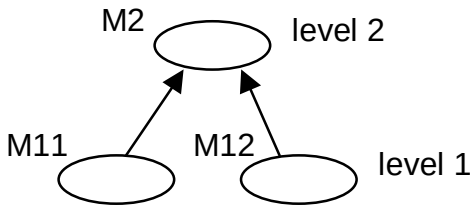

Figure 7.1. A simple two-level parallel APNN scheme. The higher-level module M1 is connected with two lower-level modules M11 and M12 by the bundles of projective connections. The internal structure of each module (an associative field and a number of buffer fields connected with each other by bundles of projective connections) is not shown.

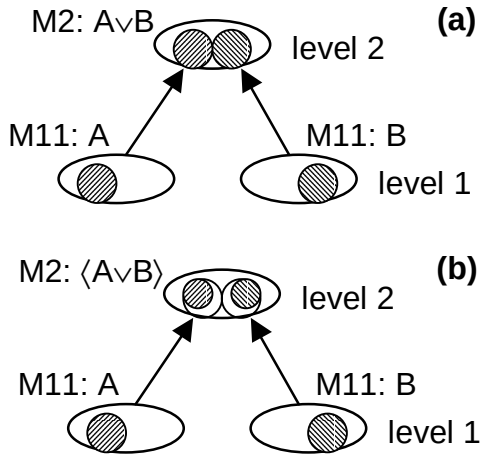

Figure 7.2. A composite two-component codevector before (a) and after (b) thinning. The components ***A*** and ***B*** are stored in M11 and M12 respectively. (a) Composite codevector ***A*** v ***B*** is formed in M2. (b) After thinning it becomes <***A*** v ***B***>. Remember that the components are really encoded not by compact, but by stochastically generated subsets of 1s.

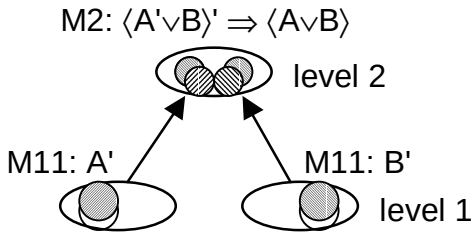

Figure 7.3. Retrieval of the closest match codevector <**AvB**> provided with the probe <**A'vB'**>.

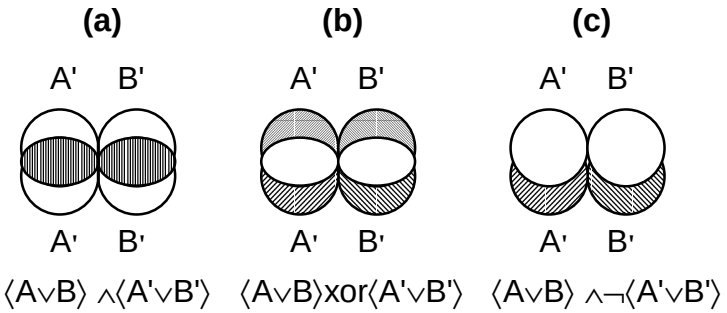

Figure 7.4. Element-wise operations with distributed codevectors (a) Similarity content (overlap), (b) symmetric difference, (c) asymmetric difference

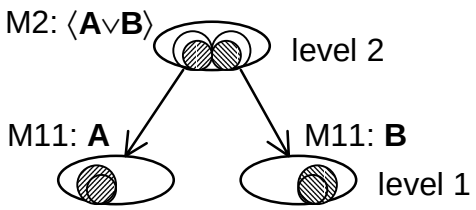

Figure 7.5. Decoding a reduced composite codevector

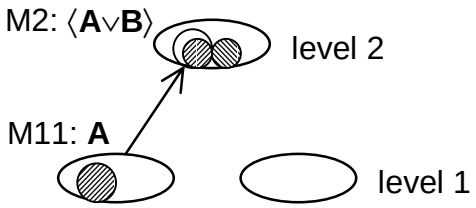

Figure 7.6. Retrieval of the whole by its part

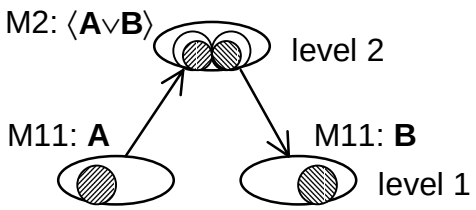

Figure 7.7. Association of two codevectors

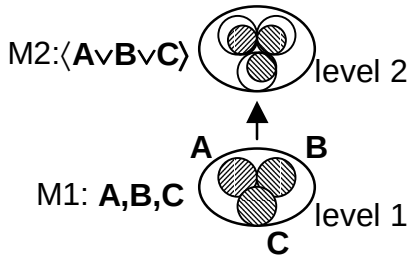

Figure 7.8 The serial APNN scheme

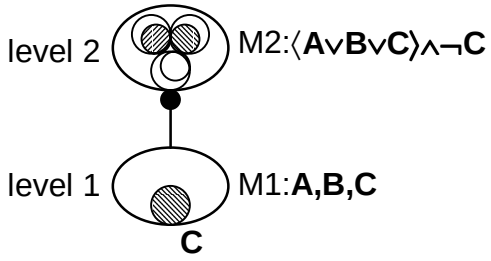

Figure 7.9 The serial APNN scheme. The already retrieved component ***C*** inhibits its representation in the reduced composite codevector <***A∨B∨C***> remaining in M2: <***A∨B∨C***>&~***C.***

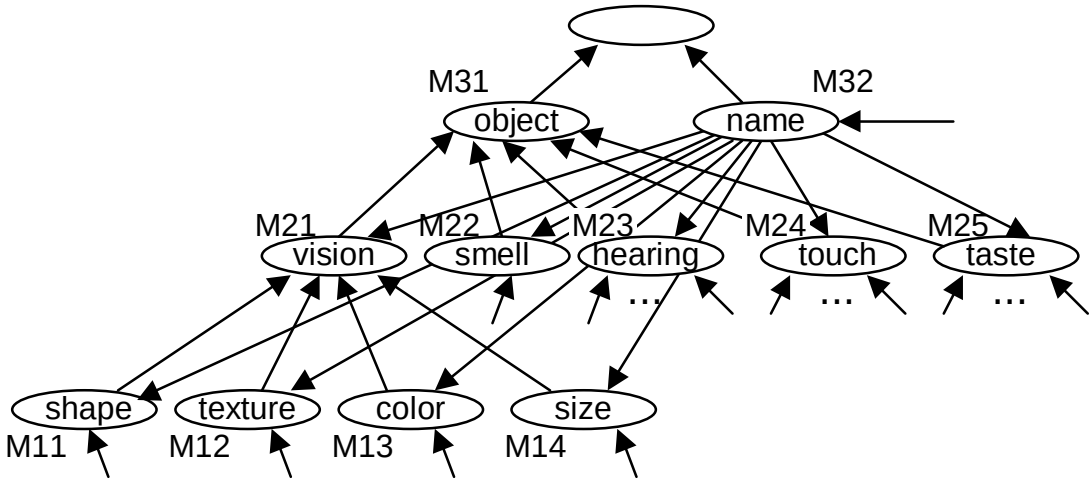

Figure 7.10. A toy example of the APNN architecture for sensory representations (models) of objects.

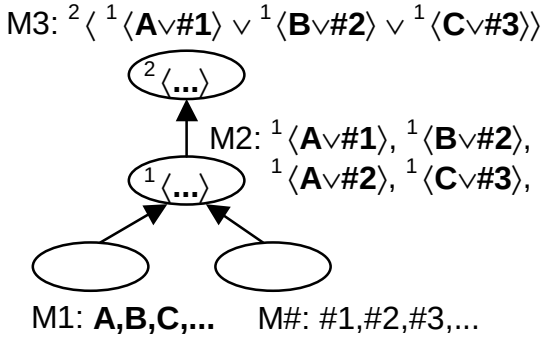

Figure 8.1. Representation of order (sequence) by binding with a role using auto-thinning

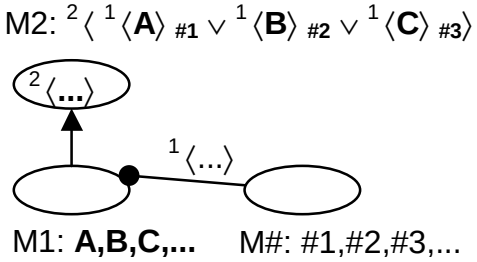

Figure 8.2. The scheme of sequence processing using hetero-thinning with the codevectors of item's ordinal numbers

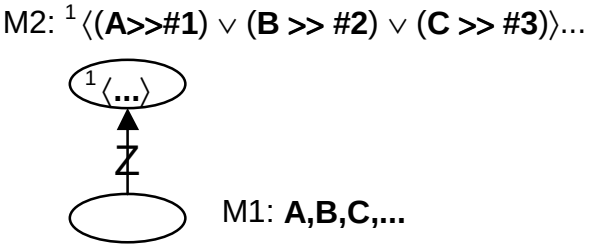

Figure 8.3. The scheme for processing of sequences represented by self-modification of item codevectors.

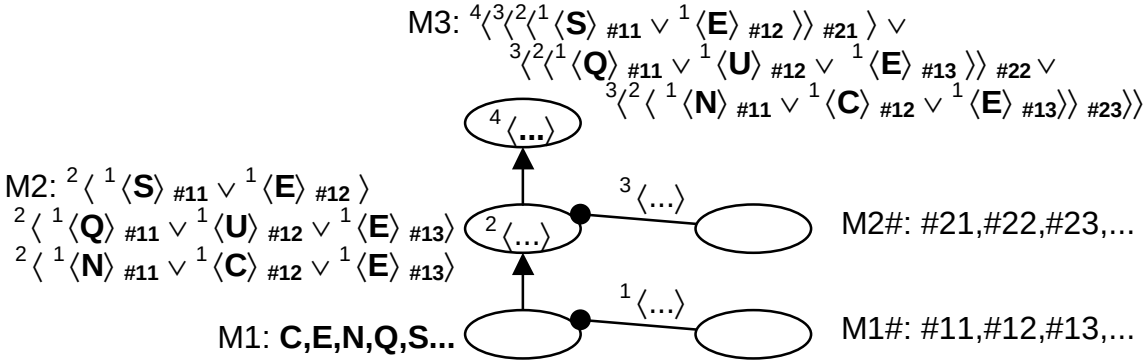

Figure 8.4. An example of the APNN architecture for processing of letter sequences. M1 - the letter module, M1# - the module of ordinal number of letters in syllables, M2 - the syllable module, M2# - the module of ordinal number of syllables in words, M3 - the word module. The word SE-QUE-NCE is processed.

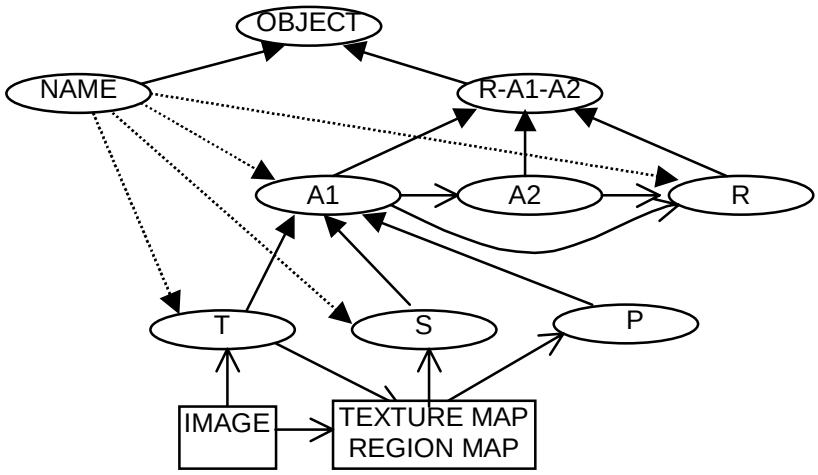

Figure 8.5 The APNN structure for visual object recognition