Abteilung Neuroinformatik, Prof.Dr.G.Palm

Associative Computation

Dissertation zur Erlangung des Doktorgrades
Dr.rer.nat der Fakultät der Informatik der Universität Ulm

von

Andreas Miroslaus Wichert

aus Wrocław

2000

Amtierender Dekan:	Prof. Dr. Uwe Schöning
Erster Gutachter:	Prof. Dr. Günther Palm
Zweiter Gutachter:	Prof. Dr. Friedrich von Henke
Externer Gutachter:	Prof. Dr. Miroslav Kubat
Tag der Promotion:	18. Juli 2000

Abstract

This work examines how human problem solving can be modeled by neural networks. Within cognitive science there are two approaches of modeling cognition: the symbolical and the pictorial or vectorial approach. In the symbolical approach cognition is viewed as symbol manipulation. Symbolical representation offers the advantage of flexible representation of structural knowledge, which is essential for problem solving. Because of this, symbolical representation is mainly used for modeling problem solving. However, symbolical representation has several weaknesses: an inability to deal with information that is fuzzy, probabilistic or noisy; a lack of robustness when processing information; difficulties to perform work in parallel. Symbolical representation is also particularly weak at learning from experience. These weaknesses of the symbolical approach are, on the other hand, advantages of the vectorial representation used by neural networks. However, vectorial representation is weak in the area of representation of structural knowledge. Consequently vectorial representation is primarily used in the domain of pattern recognition, rather than in the domain of problem solving.

To answer the question of how the human brain, which is composed of real neural networks, can solve problems, one can either try to map the symbolical approach to neural architectures, or try to build a bridge between the symbolical and vectorial approach used by neural networks. This bridge must offer flexibility of symbol manipulation and the advantages of computation in geometrical space. Categorial representation fuses the properties of symbolical and vectorial representation, providing a structure of knowledge which allows its manipulation, as well as containing a holistic vectorial representation of knowledge which enables the principle of similarity. Categorical representation offers the linkage between both approaches and enables the modeling of human problem solving behavior with artificial neural networks, which are models of the real neural networks of the human brain.

An important symbolical model of human problem solving is the production system. It is composed of a set of rules which models human long term memory, and a working memory which contains a description of the state in a problem solving process and which models human short term memory. States are described symbolically. Whenever a rule is applicable, it changes the contents of the working memory. A problem is described by the rules in the long term memory, by the

initial state, and by the desired state. The solution to the problem is represented by a chain of the rules which successively change the state from the initial state to the desired state. To find this solution a search for the right sequence of rules is performed.

Neural networks model associations in which a mapping from one vector space to another is performed. By usage of categorical representation which allows the representation of structural knowledge, it is possible to describe the transition between such represented states by associations. In this case, a problem is described by the associations in the long term memory, by the initial state, and by the desired state. The solution to the problem is represented by a chain of the associations which successively change the state from the initial state to the desired state. The associative computer model is a new model which is able to successfully determine the correct chain of associations which represent the solution to the problem.

The associative computer distinguishes itself from the pure production system in to ways.

- First, because it offers not only the hypothesis of how human problem solving is performed, but also the additional hypothesis of how this is actually performed in the real human brain.
- Secondly, it distinguishes itself significantly from the symbolical approach by additional abilities which result from the usage of neural networks and distributed representation, abilities which improve the algorithmic behavior of the model.

The model of the associative computer is developed in evolutionary steps. Simpler models are examined first and after their examination, they are integrated into more complex models. The behavior is examined empirically.

First, the neural associative memory is examined, because it is the basic building block of the succeeding models. An initial model composed of associative memories which performs associative categorization is introduced. Already even in this uncomplicated application, the main advantages of associative computation principles are demonstrated, namely fast computation linked with toleration of errors.

Categories are shown to be the basic building blocks of knowledge representation when human problem solving is simulated by neural networks. Distributed representation and similarity are the natural properties of categorical representation and it is these properties which distinguish categorical representation from symbolical representation. Verbal categories are represented by features of different salience. The quality criterion represents a rating in the presence of a category. Visual categories represent the state of the world by pictures. Similarity is used when uncertain knowledge is represented, without the need of an additional calculus. A heuristic function which can speed up the search during the problem solving stage also results from the similarity principle of categorical representation.

A neural model for a deduction system based on the assembly theory is introduced. The system consists of several neural associative memories which are organized in a hierarchical way. The problem space is represented by static connections between them. The knowledge is represented by a taxonomic arrangement of verbal categories. An availability heuristic is defined which models the effects of priming emphasis and forgetting. An availability heuristic offers a combination of the frequency with the actual likelihood of the presence of a verbal category. The model learns to favor those categories which often lead to a successful goal and this may help to speed up the search. Three applications are presented. The representation and access of large knowledge bases is demonstrated by the system Jurassic composed of 423 rules arranged in a directed acyclic graph of a depth of five. The system helps the paleontologist to determine creatures from uncertain knowledge. Experiments with the availability heuristic in which parts of the knowledge base are primed or forgotten are performed.

A neural model for a reaction system based on the biological and psychological assembly theory is introduced, the associative computer. The associative computer uses picture representation rather than symbolic representation to perform planning. Picture representation allows for the presence of noise and also enables learning from examples. The introduced permutation associative memory recognizes and executes associations. It solves two important problems which arise in the domain of neural networks. First, the traditional associative memory can not learn several possible associations which arise from a single input. The second problem is the binding problem which determines how to connect all separated fragments of a complex object. Picture representation leads to the pattern heuristic which speeds up the search. The intuitive idea that a state represented as a picture is nearer to a desired state represented also as a picture, the more similar those pictures are, is confirmed statistically. The solved problems are used to significantly speed up the search of related problems. Due to picture representation, learning from examples can be performed. The information supplied by sensors during problem solving can often be "noisy" or incomplete. Noise can also result from faulty hardware. For example, the cameras of a robot can supply noisy pictures, such as when the sensors of a space probe have a malfunction. In this case, it is important that the computing system does not collapse. It is shown that the associative computer is a robust model and can tolerate noise to some extent.

A model which attempts to explain the process of human problem solving must not be constrained to only one area. It should have the same behavior in different domains. Several exemplary domains which are well known and extensively studied in Artificial Intelligence are examined. It is shown that the used representation of the associative computer is not constrained to a special domain through examples of three different domains.

The human ability to process images and understand what they mean in order to solve a problem holds an important clue to how the human thought process works. This clue is examined by empirical experiments with the associative computer. One general conclusion from the experiments is the claim that it is possible to use systematically associative structures to perform reasoning by forming chains of associations. In addition, besides symbolical problem solving, pictorial problem solving is possible. It is, however, often helpful to use picture representation which orients itself on the *real world* when human behavior is simulated. This predication is motivated by interpretation of uncertainty with noise in pattern representation and by the three theses which sustain themselves by the experiments presented in this work.

Thesis 1 *Shape similarity between representation of states through visual categories is significantly similar to the distance in the problem space.*

Thesis 2 *Representation of states through visual categories enables the access of knowledge which was formed by learned experience during problem solving.*

Thesis 3 *An adequate model of human thought uses the flexibility of similarity-based inference and the compositionality and certainty of rule-based inference.*

A coherent research program is presented which resulted from the synthesis of the methods of Artificial Intelligence and Neurocomputation. This research program could be used as a basis for further examination.

Zusammenfassung

In dieser Arbeit wird untersucht, wie man das menschliche Problemlösen mit neuronalen Netzen simulieren kann. Innerhalb der Kognitionswissenschaft gibt es zwei Ansätze, mit denen man die kognitiven Leistungen des Menschen modelliert: den symbolischen Ansatz und den bildlichen oder vektoriellen Ansatz.

Der symbolische Ansatz identifiziert die kognitiven Leistungen des Menschen mit der Manipulation von Symbolen. Die symbolische Repräsentation bietet den Vorteil einer flexiblen Darstellung des strukturierten Wissens, welche eine notwendige Voraussetzung für das Problemlösen ist. Sie wird deshalb vorwiegend zur Modellierung des Problemlösens verwendet. Die symbolische Repräsentation zeigt jedoch einige Schwächen: z.B. beim Umgang mit ungenauer oder verrauschter Information sowie beim Lernen aus Erfahrung. Die symbolische Repräsentation ist auch weniger geeignet für eine robuste und parallele Informationsverarbeitung.

Diese Schwächen sind die Stärken der vektoriellen Repräsentation, wie sie von den neuronalen Netzen verwendet wird. Die vektorielle Repräsentation zeigt jedoch Schwächen bei der Repräsentation des strukturierten Wissens und deshalb wird sie vor allem in dem Gebiet der Mustererkennung und nicht in dem Gebiet des Problemlösens verwendet.

Um die Frage zu beantworten, wie das menschliche Gehirn, das aus neuronalen Netzen aufgebaut ist, Probleme lösen kann, könnte man entweder versuchen, den symbolischen Ansatz auf die neuronalen Netze abzubilden, oder eine Verbindung zwischen dem symbolischen und dem vektoriellen Ansatz, der von neuronalen Netzen verwendet wird, zu finden. Diese Verbindung sollte die Flexibilität der Manipulation der Symbole und die Vorteile des Rechnens in einem geometrischen Raum beinhalten. Die kategorielle Repräsentation verbindet die Eigenschaften der symbolischen und der vektoriellen Repräsentation, sie liefert sowohl eine Strukturierung des Wissens, so das es manipuliert werden kann, als auch eine ganzheitliche vektorielle Repräsentation des Wissens, welche das Prinzip der Ähnlichkeit impliziert. Die kategorielle Repräsentation ermöglicht durch die Verbindung zwischen den beiden Ansätzen die Modellierung des menschlichen Problemlösens mit künstlichen neuronalen Netzen.

Ein Produktionssystem ist ein wichtiges Modell des menschlichen Problemlösens. Es ist zusammengesetzt aus der Menge der Regeln, die das menschliche Langzeitgedächtnis modellieren, sowie dem Arbeitsspeicher, welcher die Be-

schreibung des Zustandes während des Problemlösens enthält und das menschliche Kurzzeitgedächtnis modelliert. Zustände werden mit Hilfe von Symbolen dargestellt. Immer wenn eine Regel ausführbar ist, verändert sie den Inhalt des Arbeitsspeichers. Ein Problem wird durch die Regel im Langzeitgedächtnis, sowie durch den Anfangszustand und den gewünschten Zustand, beschrieben. Die Lösung eines Problems wird durch eine Folge von Regeln dargestellt, die den Anfangszustand nacheinander in den gewünschten Zustand überführen. Um diese Lösung zu finden, wird eine Suche nach dieser Folgensequenz durchgeführt.

Neuronale Netze modellieren Assoziation durch eine Abbildung von einem Vektorraum zu einem anderen. Durch kategorielle Repräsentation, die die Repräsentation des strukturierten Wissens ermöglicht, kann man die Übergänge zwischen den so repräsentierten Zuständen durch Assoziationen beschreiben. In diesem Fall wird ein Problem durch Assoziation im Langzeitgedächtnis, sowie durch den Anfangszustand und den gewünschten Zustand beschrieben. Die Lösung eines Problems wird durch eine Folge von Assoziationen dargestellt, die den Anfangszustand nacheinander in den gewünschten Zustand überführen.

Der assoziative Computer ist ein neues Modell, welches eine Folge von Assoziationen, die die Lösung eines Problems darstellt, erfolgreich bestimmen kann. Der assoziative Computer unterscheidet sich von einem reinen Produktionssystem in zwei Punkten.

- Erstens liefert er zusätzlich zu der Hypothese, die aussagt, wie das menschliche Problemlösen durchgeführt wird, eine zusätzliche Hypothese drüber wie dieses tatsächlich im menschlichen Gehirn durchgeführt wird.

- Zweitens unterscheidet sich der assoziative Computer hauptsächlich von dem symbolischen Ansatz durch die zusätzlichen Fähigkeiten, die aus der Verwendung der neuronalen Netze und der verteilten Repräsentation resultieren: Diese Fähigkeiten verbessern das algorithmische Verhalten des Modells.

Das Modell des assoziativen Computers wird durch einen evolutionären Ansatz entwickelt. Einfachere Modelle werden zuerst untersucht, danach werden sie in komplexere Modelle integriert. Das Verhalten wird dabei empirisch untersucht.

Zuerst wird der neuronale Assoziativspeicher untersucht, weil er der Grundbaustein der nachfolgenden Modelle ist. Das erste Modell, welches eine assoziative Kategorisierung durchführt und welches aus neuronalen Assoziativspeichern aufgebaut ist, wird vorgestellt. Bereits diese einfache Anwendung demonstriert einige der wichtigsten Vorteile des Ansatzes der assoziativen Berechnung, nämlich die schnelle Berechnung verbunden mit der Korrektur von Fehlern.

Falls das menschliche Problemlösen mit neuronalen Netzen simuliert wird, stellen die Kategorien die Bausteine der Wissensrepräsentation dar. Die verteilte Darstellung und die daraus resultierende Ähnlichkeit sind wichtige Eigenschaften der kategoriellen Repräsentation, und es sind diese Eigenschaften, die sie von der symbolischen Repräsentation unterscheiden. Verbale Kategorien werden durch Merkmale unterschiedlicher Gewichtung repräsentiert. Das Qualitätskriterium stellt ein Vertrauensmaß in das Vorhandensein einer Kategorie dar. Visuelle Kategorien repräsentieren die Zustände der Welt durch Bilder. Ähnlichkeit wird verwendet, um das unsichere Wissen darzustellen, ohne daß eine zusätzliche Rechenvorschrift, die die Unsicherheit separat behandelt, benötigt wird. Aus dem Ähnlichkeitsprinzip der kategoriellen Repräsentation ergibt sich auch eine heuristische Funktion, die die Suche während des Problemlösens entscheidend beschleunigen kann.

Ein Modell eines neuronalen Deduktionssystems, welches auf der Assemblie-Theorie beruht, wird vorgestellt. Das System besteht aus einigen hierarchisch angeordneten neuronalen Assoziativspeichern. Der Problemraum wird durch die Verbindungen zwischen den neuronalen Assoziativspeichern repräsentiert. Wissen wird durch die taxonomisch angeordneten verbalen Kategorien repräsentiert. Die Verfügbarkeitsheuristik wird definiert, die die Effekte des Priming, der Voraktivierung und des Vergessens des Wissens modelliert. Die Vefügbarkeitsheuristik vereint in sich die Wahrscheinlichkeit für das aktuelle Vorhandensein einer verbalen Kategorie und die Häufigkeit ihres vorherigen Vorhandenseins. Das Modell, lernt die Kategorien zu bevorzugen, die oft erfolgreich zum Ziel führten. Dieses kann die Suche beschleunigen. Drei Anwendungen werden vorgestellt. Die Repräsentation und der Zugriff auf das Wissen einer großen Wissensbasis wird durch das System Jurassic demonstriert, welches aus 423 Regeln, die in einem azyklischen Graph der Tiefe fünf angeordnet sind, aufgebaut ist. Das System hilft einem Paläontologen, Spezies mit unsicherem Wissen zu bestimmen. Experimente mit der Verfügbarkeitsheuristik, in welchen Teile der Wissensbasis voraktiviert oder vergessen werden, werden durchgeführt.

Mit dem assoziativen Computer wird ein neuronales Modell eines Reaktionssystems vorgestellt, basierend auf der biologischen und physiologischen Assemblie-Theorie. Dieser assoziative Computer verwendet eine bildhafte Repräsentation anstatt der symbolischen Repräsentation, um Planungsaufgaben durchzuführen. Die bildhafte Repräsentation erlaubt das Vorhandensein von Rauschen, zusätzlich wird das Lernen aus Erfahrung ermöglicht. Der eingeführte Permutationsassoziativspeicher führt erkannte Assoziationen aus. Er löst zwei wichtige Probleme der Neuroinformatik. Das erste Problem besagt, daß gewöhnliche Assoziativspeicher nicht unterschiedliche Assoziationen lernen können, die zu einer bestimmten Eingabe gehören. Das zweite Problem ist das Bindungsproblem, welches bestimmt, wie unterschiedliche Teile miteinander verbunden werden sollen, um ein Objekt zu repräsentieren. Bildhafte Repräsentation führt zu einer Musterheuristik, die die Suche beschleunigt. Wenn man einen Zustand durch ein Bild repräsentiert, dann besagt eine intuitive Idee, daßzwei verschiedene Zustände umso näher aneinanderliegen, je ähnlicher sich ihre beiden Bilder sind. Diese Idee wird mit Hilfe statistischer Untersuchungen bestätigt. Das Wissen über gelöste Probleme wird verwendet, um die Suche ähnlicher Probleme signifikant zu beschleunigen. Durch die bildhafte Repräsentation wird nämlich das Lernen aus Erfahrung ermöglicht. Falls Sensoren Informationen liefern, die während des Problemlösens benötigt werden, können diese Informationen oft verrauscht oder unvollständig sein. Rauschen ergibt sich zur Zeiten durch fehlerhafte Bauteile, so können die Kameras eines Roboters verrauschte Bilder liefern oder die Sensoren einer Raumsonde können teilweise defekt sein. Es ist wichtig, daß das Rechnersystem in diesem Fall nicht zusammenbricht. Es wird gezeigt, daß der assoziative Computer ein robustes Modell ist, welches Rauschen bis zu einem bestimmten Grad toleriert. Falls man das menschliche Problemlösen erklären will, darf ein dazugehöriges Modell nicht nur auf eine Anwendung eingeschränkt sein. Das Modell soll in der Anwendung auf unterschiedlichen Gebieten das gleiche Verhalten aufweisen. Es werden Beispiele aus sehr bekannten Gebiete der Künstlichen Intelligenz untersucht. Durch Anwendungen aus drei unterschiedlichen Gebieten wird gezeigt, daß die verwendete Repräsentation nicht nur auf ein bestimmtes Gebiet eingeschränkt ist.

Die menschliche Fähigkeit, Bilder in Zusammenhang mit ihrer Bedeutung für das Prolemlösen zu verstehen, beinhaltet einen wichtigen Wegweiser, der zeigt wie das menschliche Denken aufgebaut ist. Diesem Wegweiser wird durch empirische Experimente mit dem assoziativen Computer nachgegangen. Ein allgemeines Ergebnis dieser Experimente ist die Aussage, daß durch den systematischen Gebrauch assoziativer Strukturen das Schlußfolgern durch die Bildung von Ketten aus Assoziationen durchgeführt werden kann. Zusätzlich zu dem symbolischen Problemlösen ist das bildhafte Problemlösen möglich. Bei der Simulation des menschlichen Problemlösens ist es oft hilfreich eine bildliche Repräsentation zu verwenden, welche sich an der wirklichen Welt orientiert. Diese Aussage ist durch die Interpretation der Unsicherheit mit dem Rauschen in der bildhaften Repräsentation und durch die drei Thesen motiviert, die sich auf die Ergebnisse der Experimente dieser Arbeit stützen.

These 1 *Die Ähnlichkeit, die sich aus der Form der Zustände ergibt, die durch visuelle Kategorien repräsentiert werden, entspricht signifikant häufig ihrer Entfernung in dem Problemraum.*

These 2 *Die Repräsentation der Zustände durch visuelle Kategorien ermöglicht den Zugriff auf das Wissen, welches aus der Erfahrung des Problemlösens stammt.*

These 3 *Ein angemessenes Modell des menschlichen Denkens verwendet die Flexibilität des ähnlichkeitsbasierenden Folgerns und die Sicherheit des regelbasierenden Folgerns.*

Ein abgeschlossenes Forschungsprogramm wird dargestellt, welches sich aus der Synthese der Methoden der Künstlichen Intelligenz und der Neuroinformatik ergibt. Es kann als das Fundament für weitere Untersuchungen angesehen werden.

Résumé

Ce mémoire étudie comment il est possible de modéliser la capacité humaine de la résolution de problèmes avec des réseaux de neurones. Dans les sciences cognitives il y a deux approches de modéliser la performance cognitive d'un homme: une approche symbolique et une approche image ou vectorielle. L'approche symbolique identifie la performance cognitive avec la manipulation de symboles. La représentation symbolique a l'avantage d'une flexible description de connaissances structurées. C'est une condition indispensable pour la résolution de problèmes. Pour cette raison la représentation symbolique est utilisée surtout pour modéliser la résolution de problèmes. La représentation symbolique a pourtant des faiblesses: Des faiblesses quand une information inexacte, accidentelle ou bruitée est présente; la possibilité de modification par l'apprentissage; le manquant de la robustesse dans le traitement de l'information; le manquant de traitement parallèle de l'information. Les manquants d'une approche symbolique sont d'autre part les atouts d'une approche vectorielle utilisé par les réseaux de neurones. La représentation vectorielle a pourtant des manquants dans la représentation de connaissances structurées. Donc la représentation vectorielle est utilisée essentiellement dans la reconnaissance des formes.

Pour répondre à la question comment le cerveau humain, qui est composé des réseaux de neurones, résoud des problèmes, on pourrait essayer de reproduire l'approche symbolique par les réseaux de neurones, ou de trouver une liaison entre la représentation symbolique et la représentation vectorielle utilisée par des réseaux de neurones. Cette liaison doit contenir une flexibilité de manipulation de symboles et les avantages d'un calcul dans l'espace géométrique. La représentation catégorielle réunit les qualités de la représentation symbolique et de la représentation vectorielle, livrant la structure de connaissances qui permettent leur manipulation, mais également une représentation vectorielle globale de connaissances qui permet le principe de la similarité. Par liaison de ces deux approches, la représentation catégorielle permet de modéliser la capacité humaine de résolution de problémes avec des réseaux de neurones artificielles, qui sont des modèles des réseaux de neurones de cerveaux.

Le système de production est un modèle important de capacité humaine de résolution de problèmes. Il est composé d'un ensemble de règles, qui modélisent la mémoire humaine longterme, ainsi que la mémoire vive avec la description

de l'état pendant la résolution de problèmes, qui modélise la mémoire à court terme. Les états sont représentés par des symboles. Chaque fois une règle est exécutable, elle change le contenu de la mémoire vive. Un problème est décrit par les règles dans la mémoire à long terme, par l'état initial et par l'état désiré. La solution d'un problème est représenté par une séquence de règles qui transforment l'état initial dans l'état désiré. Pour trouver la solution une recherche pour cette séquence a lieu.

Les réseaux de neurones modélisent les association par une transformation d'un espace vectoriel dans un autre. Parce que une structuration de connaissance est possible avec la représentation catégorielle, il est possible de décrire des transitions entre états avec des associations . Dans ce cas, un problème est décrit par les associations dans la mémoire à long terme, par l'état initial et par l'état désiré. La solution d'un problème est représenté par une chaîne d' associations qui transforment l'état initial dans l'état désiré. L'ordinateur associatif est un modèle nouveau qui détermine avec succès la chaîne d'associations qui représentent la solution d'un problème. L'ordinateur associatif se distingue en deux points du système de production.

- En premier lieu il livre une hypothèse supplémentaire à l'hypothèse qui explique comment les hommes résoudent des problèmes, l'hypothèse qui explique comment actuellement le cerveau humain résoud des problèmes.

- En second lieu l'ordinateur associatif se distingue essentiellement d'une approche symbolique par des facultées additionnelles, qui résultent de l'utilisation de réseaux de neurones et d'une représentation distribuée. Facultées qui améliorent la conduite algorithmique.

Le modèle d'ordinateur associatif est développé par une approche évolutive. D'abord les modèles simples sont examinées, après ça ils sont intégrés dans des modèles plus complexes. La conduite est examinée par une approche empirique.

D'abord la mémoire associative de neurones est examinée, parce qu'elle est un élément de base des modèles suivants. Le premier modèle composé de mémoires associatives, qui réalise une catégorisation associative, est introduit. Déjà cette utilisation facile démontre plusieurs avantages importants d'une approche du calcul associatif, calcul rapide avec correction des fautes.

Les catégories représentent les pierres de construction au cas où la résolution humaine de problèmes avec les réseaux de neurones est modélisée. La représentation distribuée et la similarité sont des qualités importantes d'une représentation catégorielle. Ce sont ces qualitées qui la distinguent d'une représentation symbolique. Les catégories verbales sont représentées par des caractéristiques de différente force. Le critère de qualité représente une mesure de confiance de la présence d'une catégorie. Des catégories visuelles représentent les états d'un monde par des images. La similarité est utilisée pour représenter des connaissances incertaines, sans utiliser un calcul additionnel qui représente cette insécurité. Le principe de similarité de la représentation catégorielle est aussi une fonction heuristique, qui accélère la recherche.

Un modèle d'une système da déduction neuronale inspiré par la "assembly" théorie est introduite. Le système est composé de plusieures mémoires associatives de neurones arrangées hiérarchiquement. L'espace de problèmes est représenté par les liaisons entre les mémoires associatives. Les connaissances sont représentées par des catégories verbales qui sont arrangées dans une taxonomie. La définition heuristique de disponibilité exprime la préactivation ("priming") et l'oubli. L'heuristique de disponibilité réunie la probabilité de la présence d'une catégorie verbale avec la probabilité antérieure de la présence de cette catégorie verbale. Le modèle apprend à préférer des catégories qui étaient souvent couronné de succès. Cette préférence accélère souvent la recherche. Trois applications sont introduites. La représentation et l'accès a une grande base de connaissance est démontrées par le système Jurassique, qui est compose de 423 règles ordonnées dans un graphe acyclique de profondeur cinq. Le système serve au paléontologue pour déterminer une espèce à partit de connaissances incertaines. Expériences avec l'heuristique de disponibilité avec la base de connaissance préactive ou oublie sont réalisées

Un modèle d'un système da la réaction neuronale inspiré par la biologie et psychologie "assembly" théorie est introduite. C'est l'ordinateur associatif. L'ordinateur associatif utilise la représentation avec des images et non pas une représentation symbolique pour sauver des problèmes de la planification. La représentation avec des images permet la présence d'une information inexacte ou bruitée et la possibilité de modification par apprentissage. La mémoire associative de la permutation exécute des associations. Elle résoud deux problèmes importants des réseaux de neurones. Premier: La mémoire associative traditionelle n'est pas capable d'apprendre différentes associations pour une entrée. Le deuxième problème est le problème du lien, de déterminer comment joindre les différentes parties de l'objet. La représentation avec des images conduit à une heuristique d'images, qui accélère significativement la recherche. L'idée intuitive que l' état représenté avec une image est plus proche a un état désiré représenté également par une image, quand les deux images se ressemblent, est confirmé statistiquement. La connaissance de la solution de problèmes est utilise pour trouver une solution plus vite pour d'autres problèmes similaires. Avec une représentation qui utilise des images l'apprentissage est possible. Des informations sensorielles sont souvent inexacte ou bruitée. Les pièces sont souvent plein de fautes, les cameras d'un robot donnent des images avec du bruit, les senseurs d'une sonde spatiale sont en panne. C'est important en ce cas qu'un calculateur automatique ne s'effondre pas. On montre que l'ordinateur associatif est un modèle robuste, un modèle qui tolère l'information bruitée jusqu'à un certain degré. Un modèle qui explique la capacité humaine de résolution de problèmes ne doit pas être restreint à un domaine. Le modèle doit avoir la même manière d'agir dans des domaines différents. Des exemples de domaines de l'intelligence artificielle très connus sont examinés. Par les trois applications de différents domaines on montre que la représentation n'est pas limitée à un seul domaine. La capacité humaine de comprendre les images en relation avec leurs signification pour la résolution de problèmes est un guide qui montre comment les hommes pourraient penser. Ce guide est examiné par les expériences empiriques avec l'ordinateur associatif. Inférer par les chaînes des associations est possible, c'est le premie résultat général.

En plus de la résolution de problèmes symbolique, la résolution de problèmes avec des images est possible. C'est souvent plus favorable d'utiliser la représentation orienté sur le monde véritable, quand on simule la résolution de problèmes par hommes, la représentation avec des images. Cette déclaration est motivée par l'interprétation de l'insécurité dans la représentation distribuée et par les trois thèses, qui sont supporte par les expériences de ce mémoire.

Thèse 1 *La similarité des images qui représentent des états par des catégories visuelles correspond leur distance dans l'espace de problèmes.*

Thèse 2 *La représentation des états par des catégories visuelles permet l'accès à la connaissance, qui était formée durant la résolution de problèmes.*

Thèse 3 *Un modèle convenable de la pense humain utilise la flexibilité de la similarité pendant l'inférence et la certitude de l'inférence avec des règles.*

Un domaine de recherche est préparé, construit de la synthèse des méthodes de l'intelligence artificielle et des réseaux de neurones.

Acknowledgements

It has been a very long way from original sketchy ideas to the publication of this work. I would like to thank the following people, and offer all of them deepest gratitude. I thank my adviser Prof. Dr. Günther Palm who has influenced my research in a large way. I am very grateful to his support for exploring topics far away from the mainstream. I thank Prof. Dr. von Henke and Prof. Dr. Miroslav Kubat for their valuable comments and their commitment to act as referees for this work, despite their faculty duties. I am grateful to Dr. Wilko Ahlrichs from the section for biosystematic documentation for discussions about systematics and biosystematic research databases.

I thank to all my colleagues from the Department of Neural Informatics, they made the department a fun place to work.

Finally, I would like to thank *Christiane*, whose patience and encouragement knows no bounds. My family has been an endless source of support, thanks.

Contents

Chapter 1

Introduction

1.1 Prologue

In this work we will examine how human problem solving [179] can be modeled by neural networks. Examples of the types of problems adressed include the determination of some consequence from vague and fuzzy knowledge or the outline of a sequence of actions which lead to some desired goal. Problems of this kind are solved by algorithms studied by artificial intelligence. The key idea behind these algorithms is the symbolic representation of the domain in which the problems are solved. Symbols are used to denote or refer to something other than themselves, namely to other things in the world (according to the pioneering work of Tarski [204, 205, 206]). They are defined by their occurrence in a structure and by a formal language which manipulates these structures [177, 178, 132, 133]. In this context symbols do not by themselves, represent any utilizable knowledge. For example, they could not be used for a definition of similarity criteria between themselves. The use of symbols in algorithms which imitate human intelligent behavior led to the famous physical symbol system hypothesis by Newell and Simon (1976) [135]: "The necessary and sufficient condition for a physical system to exhibit intelligence is that it be a physical symbol system."

Artifical neural networks which are simple models of real neural networks of living animals use mainly vector representation, as it mirrors the way the biological sense organs describe the world [31]. Vectors define a space in which the similarity between themselves can be computed. Because of this property, vector representation stands in contrast to the physical symbol system hypothesis [45, 36]. Consequently, neural networks are mostly used in the domain of pattern recognition [70], rather than in the domain of problem solving. Of course, it is also possible to use a symbolic representation in neural networks, but then some important properties which characterize neural networks are lost. These properties include: the ability to deal with information that is fuzzy, probabilistic or noisy; the ability of adjustment by learning; robustness and parallel work. Therefore, this work does not examine the reimplementation of symbolic algorithms in the domain of problem solving by neural networks, but rather examines the use of vector or pattern based representation and the resulting consequences.

1.2 Motivation and Goals

Currently neural networks are used in many different domains. But are neural networks also suitable for modeling problem solving, a domain which is traditionally reserved for the symbolic approach? This central question of cognitive science is answered in this work. It is affirmed by corresponding neural network models. The models have the same behavior as the symbolic models. However, also additional properties resulting from the distributed representation emerge. It is shown by comparison of those additional abilities with the basic behavior of the model, that the additional properties lead to a significant algorithmic improvement. This is verified by statistical hypothesis testing. The division of the behavior of the model into basic behavior and additional property behavior corresponds to the Michalskis two tier philosophy of concept meaning [120, 121, 89].

This work is motivated by several important theories from the fields of biology, computer science and psychology. These theories include the biological neural assembly theory, the psychological categorical representation theory, the biological and psychological mental representation theory, and the production system theory from the domain of computer science. The goal of this work is a neural problem solving model which gives some insights into how the brain solves certain problems by forming chains of associations.

Neural assembly theory, which was first suggested by Donald Hebb (1949) [65], describes the bridge between the structures found in the nervous system and in a high level cognition such as problem solving. An assembly of neurons acts as a closed system, and therefore can represent a complex object. Activation of some neurons of the assembly leads to the activation of the entire assembly, so that manipulations on the representation of a complex object is performed [66, 2, 52, 146]. These complex objects are correspond to thoughts. The process of problem solving is described as the transformation of thoughts by a group of assemblies [24, 143]. Thoughts are propagated from one assembly to another until a desired thought, which represents a goal, is reached. The “Associative memory” is a formal neural net model of this assembly concept [143]. It is formed from a group of neurons. With these neurons patterns can be stored in such a way that, when a new pattern is presented, a pattern is formed which closely resembles the stored patterns by the activation of the whole corresponding assembly. By forming associations, an associative memory can link together patterns which are represented by vectors. The idea of describing the human reasoning process as the formation of associations is an old idea which can be traced back to Aristotle (de Memoris et Reminiscenta) [11]. However, even now at the very beginning of the 21st century, research on complex associative systems is still at the forefront of research in cognitive science [7].

Thoughts can be understand as the description of complex objects. These complex objects are structured and formed by different fragments which can be represented by categories according to cognitive psychology [185]. Categorical representation offers an explanation of how to deal with similarity between objects. There are two different types of categories: verbal categories and visual categories. It is a straight forward matter to represent verbal categories by vectors. Vector representation of complex visual categories, however, must adress the "binding problem". This is the problem of maintaing the structure necessary for the formation of correlation between different fragments. The biological "what" and "where" system [88, 156] suggests a solution to this issue.

Productions systems theory describes how to form a sequence of actions which lead to a goal, and offers a computational theory of how humans solve problems [133, 229]. Production systems are composed of if-then rules which are also called productions. The complete set of productions constitute long term memory. Productions are triggered by specific combinations of symbols which describe items. These items represent a state and are stored in short term memory. A computation is performed with the aid of productions by the transformation from an initial state in the short term memory to a desired state.

This complex of ideas led the direction of preceding research to the neural network associative computer model. This model solves problems by forming a chain of associations. The associations are stored in a new assembly concept model, the "permutation associative memory", which also solves the binding problem during problem solving. To allow for learning from experience, an additional associative memory is used. Learning from experience and the information which give hints as to which associations should be used (heuristics) result from the visual categorical representation of the problems. This information is lost if symbolic representation is used instead, and a programmer must append it to the architecture to achieve the same informed behavior. This additional step is not always an easy task due to the lack of a corresponding formal method.

Two examples of problems which are solved by the associative computer from the robot domain follow:

The first example consists of the task of building a tower from a collection of blocks [136]. A robot arm can stack, unstack, and move the blocks within a plane on eight different positions at a table. There are two different classes of blocks: cubes and pyramids. While additional blocks may be stacked on top of a cube, no other blocks may be placed on top of a pyramid. The robot arm, which is represented in the upper right corner, has a gripper that can grasp any available block. It can move the block to eight different positions on the tabletop or place it on top of another cube. The states are represented by patterns which correspond to two-dimensional visual categories. A sequence of the plan is shown in fig. 1.1.

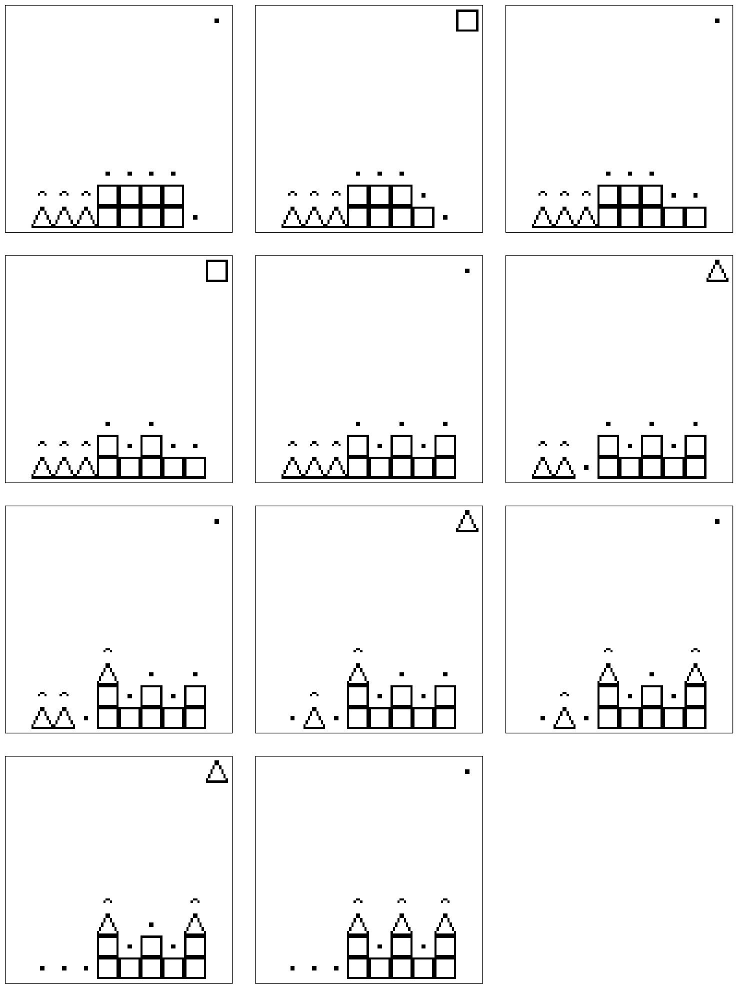


Figure 1.1: Planning of the tower building task, guided by similarity of resulting patterns.

Similarity of resulting patterns to the desired state description pattern of the tower guides the choice of the robot arm actions. Thus, information which guides the robot arm actions is represented in the patterns and would be lost in a symbolical representation of the states.

The second example shows the world of the robot Clotaire, a robot which can move from one position to another [43]. One can represent this state symbolically, for example as position (Clotaire,2), north(1,2), south(2,1), east(1,4). For such a representation, noise which results from the sensorial map description of the labyrinth must first be eliminated. The associative computer can deal with noise in the state description by patterns, as shown in the pattern representation of Clotaire's tour from *Hangar* to *Atelier* (fig. 1.2).

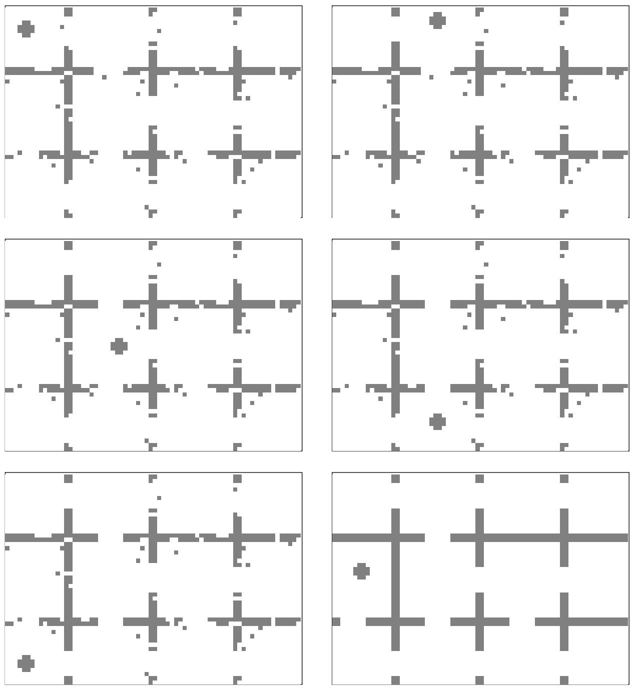


Figure 1.2: Planning of the robot task to go from *Hangar* to *Atelier* with noise after learning.

In our example the shortest path is chosen because Clotaier had previously explored and already learned the labyrinth before.

The associative computer model is a universal neural network model which can solve many different problems, including some of the well known problems from the domain of AI. Among the examples is the "8-puzzle" problem which is difficult due to nearly "9!" (=362880) configurations [111].

The associative computer model techniques resulted from experiments on verbal categorical representation of hierarchical categorization by neural networks. The verbal categorical representation by associative memories is an alternative to other well known uncertainty calculi [76]. During hierarchical categorization one moves from a more general category to a more specific category until a goal is reached. One of the tested systems, a system from the field of paleontology for the determination of dinosaur species out of nearly 340 species, is presented.

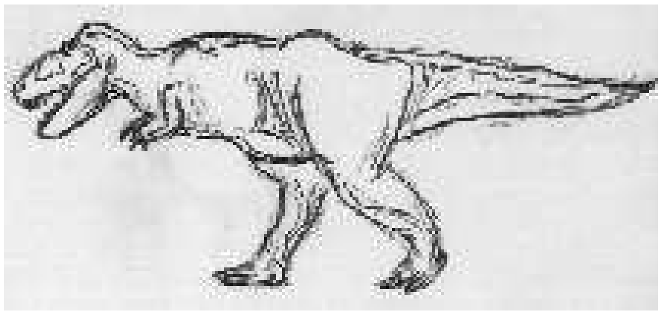


Figure 1.3: Tyranonosaurus.

In the example fig.1.3 the observer specified the species by seven features. The hierarchical categorization is performed by associative memories until no match of the features to a verbal category is good enough. The match is shown by a quality criteria (qc) which is a numerical value of the range from -1 (no match) to 1 (perfect match).

1. certainly **two legged**
2. certainly not **four legged**
3. certainly **flesh eating**
4. certainly **big**
5. certainly **large head**
6. certainly **dagger like teeth**
7. very probably **length ten to fourteen m**

```
1) ! MODULE SAURISCHIAN with qc=-0.33, <1> !***
2) ! MODULE THEROPODS with qc=0.33, <2> !*
3) ! MODULE CARNOSAURS with qc=0.26, <3> !**
4) ! MODULE TYRANNOSAURIDS with qc=0.22, <4> !*
----------------------------------------------
R E S U L T :
```

TARBOSAURUS with qc=0.26
TYRANNOSAURUS with qc=0.26

```
5) ! MODULE MEGALOSAURIDS with qc=0.19, <4> !
6) ! MODULE ALLOSAURIDS with qc=0.19, <4> !
7) ! MODULE TERATOSAURIDS with qc=0.18, <4> !
8) ! MODULE SEGNOSAURIDS with qc=0.18, <4> !
9) ! MODULE SPINOSAURIDS with qc=0.17, <4> !
10) ! MODULE CERATOSAURIDS with qc=0.17, <4> !
11) ! MODULE DRYPTOSAURIDS with qc=0.16, <4> !
12) ! MODULE OTHER with qc=0.11, <4> !*
----------------------------------------------
R E S U L T :
```

DIMETRODON REPTILE with qc=0.13

```
13) ! MODULE THERIZINOSAURIDS with qc=0.11, <4> !
14) ! MODULE COELUROSAURS with qc=0.22, <3> !
...
...
40) ! MODULE IGUANODONTIDS with qc=0.08, <4> !
41) ! MODULE STAURIKOSAURIDS with qc=0.06, <3> !
42) ! MODULE STRANGE_KILLERS with qc=0.03, <2> !
43) ! MODULE ORNITHISCHIAN with qc=-0.33, <1> !**

sorry, I have no more acceptable answer for You.....
```

A concern with hierarchical categorization is the use of inexact, missing, or poorly defined information. This problem is solved in our example by the pattern representation of the verbal categories. One way which symbolic expert systems have to address this issue is to attach a numeric weight to the conclusion of each symbolic rule and to use an uncertainty calculus [76].

1.3 Empirical Methods

This work demonstrates how human problem solving can be modeled by neural networks with the aid of empirical experiments. The experiments were based on programs generating data which describe the behavior of the simulated models [33]. Hypotheses about the behavior of the models were tested by statistical analysis with collected data. The programs were implemented by a software laboratory which was built using an object oriented approach. Simpler models were developed and tested first. After their analysis, more complex models were designed using the resulting knowledge. This approach led to the reduction of problems in the design phase of the models and software, and also to a notable reduction of software errors which typically occur in such complex simulations.

1.4 Guide to the Reader

Associative Neural Memory - Chapter 2 This chapter explains the associative neural memory, which models the behavior of neural assemblies and is the basic part of the associative computation models. Its usage is demonstrated on simple models. For example, the fast associative categorization model offers an alternative approach to the storage of large amounts of data.

Representation - Chapter 3 This chapter begins with the representation formalisms which are traditionally used in AI. Predical Calculus, Frames, and Rules representation offer a way to represent knowledge symbolically. The principle of similarity is the core of the verbal- and visual-categorical representation. Both categorical neural representation formalisms and the resulting principles of similarity are demonstrated by examples.

Hierarchical Categorization - Chapter 4 Production systems are described in this chapter. They are the basic working principles of many expert systems. Based on these techniques, an initial associative computation model which offers a foundation for a more general model is introduced. The hierarchical categorization model is composed of associative memories which are organized hierarchically. Uncertain knowledge is used in this model for the determination of some desired verbal categories by moving from more general verbal categories to more specific verbal categories. During the hierarchical categorization the model learns to favor these verbal categories which often lead to a goal. This system is illustrated by a diagnostic system for the determination of the disorders of a car. The idea of usage of hierarchical categorization as an interactive instruction system is demonstrated on a small system for the diagnosis and medical treatment of reanimation. Finally, a system from the field of paleontology for the determination of dinosaur species demonstrates the learning of habits by favoring the categories which often lead to a goal. In addition, it shows the usage as an expert system in the biological field of systematics. At the end of the chapter, the drawbacks which result from the static representation of the search space by the hierarchical categorization model are described. In addition, the demand for a more flexible associative computation model is expressed.

Associative Computer - Chapter 5 The hypothesis that production systems are a model of human-problem solving behavior is the basis of the associative computation model, the associative computer. The associative computer represents the search space dynamically. It is composed of permutation associative memory which allows the representation of structured knowledge as well as of a neural search chain which represents the problem space. The behaviors of the model such as backtracking, tolerance of noise, representation defined heuristics, and learning from examples is demonstrated by empirical experiments in a geometrical block world. This block world is composed of two cubes and a pyramid which are manipulated by a robot arm.

Experiments with the associative computer - Chapter 6 The ABC block world is demonstrated in this chapter. It is shown that the use of representation defined heuristics and learning improves the behavior of the model in a statistically significant manner. The associative computer's representation of the 9 puzzle problem is shown as well as resulting learning from experience. The final experiments are on a robot in a labyrinth.

Comparison to Related works - Chapter 7 The associative computer model is compared to connectionistic models. The comparison of the connectionistic models is structured in localistic connectionistic models, distributed conectionistic models and assembly models.

Conclusion - Chapter 8 In the conclusion the results are highlighted in a compact form. At the end of the work, the ideas which may guide possible future research on associative computation and assembly theory are presented. In Appendix A.1 the statistical tools which are used in the empirical experiments are explained. Appendix A.2 illustrates the organization of the object oriented laboratory describes information concerning the implementation of the associative computer. Appendix A.3 shows some examples of heirarchical categorization.

Chapter 2

Associative Neural Memory

"Human memory is based on associations with the memories it contains. Just a snatch of well-known tune is enough to bring the whole thing back to mind. A forgotten joke is suddenly completely remembered when the next-door neighbor starts to tell it again. This type of memory has previously been termed content-addressable, which means that one small part of the particular memory is linked - associated -with the rest." Cited from [28], page 104. The advantages of associative memory compared to the random access memories arise from the following abilities [143, 70, 8, 85]:

- The ability to correct faults if false information is given.
- The ability to complete information if some parts are missing.
- The ability to interpolate information, in other words if a pattern is not currently stored the most similar stored pattern is determined.

Associative memory models human memory [144, 32, 50, 192].

2.1 The Lernmatrix

The Lernmatrix, also simply called "associative memory", was developed by Steinbuch in 1958 as a biologically inspired model from the effort to explain the psychological phenomenon of conditioning [193, 194]. Later this model was studied under biological and mathematical aspects by G. Palm [143, 144, 47, 219]. It was shown that Donald Hebb's hypothesis of cell assemblies as a biological model of internal representation of events and situations in the cerebral cortex corresponds to the formal associative memory model. We call the Lernmatrix simply "associative memory" if no confusion with other models is possible. The associative memory is composed of a cluster of units which represent a simple model of a real biological neuron (see fig. 2.1).

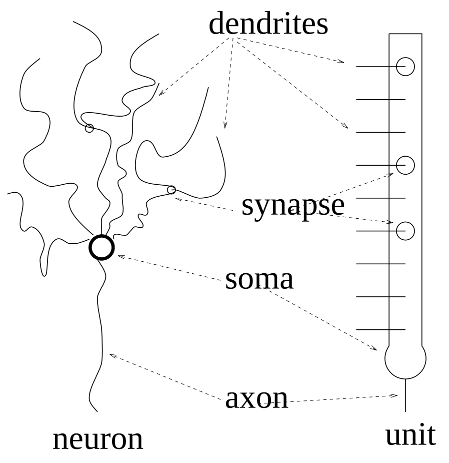


Figure 2.1: A unit is an abstract model of a biological neuron [114, 144, 70, 139, 169].

The unit is composed of weights which correspond to the synapses and dentrides in the real neuron. They are described by w_{ij} in fig. 2.2. T is the threshold of the unit.

The patterns are represented by binary vectors. The presence of a feature is indicated by a "one" component of the vector, its absence through a "zero" component of the vector. Two pairs of these vectors are always associated and this process of association is called learning. The first of the two vectors is called the question vector and the second, the answer vector. After learning, the question vector is presented to the associative memory and the answer vector difference is determined. This process is called association. [1]

2.1.1 Learning and Forgetting

In the initialization phase of the associative memory no information is stored. Because the information is represented in the weights, they are all initially set to zero. In the learning phase, binary vector pairs are associated. Let $\vec{x}$ be the question vector and $\vec{y}$ the answer vector, so that the learning rule is:

$$w_{ij}^{new} = w_{ij}^{old} + y_i x_j.$$

This rule is called the binary unclipped Hebb rule [143]. Every time a pair of binary vectors is stored this rule is used. Therefore, in each weight of the associative memory the frequency of the correlation between the components of the

[1]In the literature often a distinction between heteroassociation and association is made. An association is present when the answer vector represents the reconstruction of the faulty question vector. An heteroassociation is present if both vectors are different.

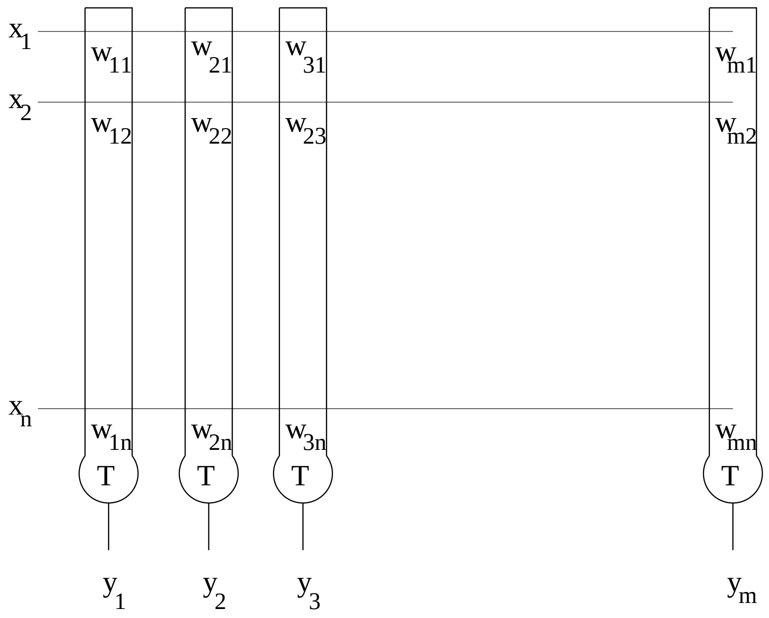


Figure 2.2: The associative memory is composed of a cluster of units [144, 149].

vectors is stored. This is done to ensure the capability to "forget" vectors which were once learned. In this case, the following binary anti-Hebb rule [215] is used:

$$w_{ij}^{new} = \begin{cases} w_{ij}^{old} - y_i x_j & \text{if } w_{ij}^{old} > 0 \\ w_{ij}^{old} & \text{if } w_{ij}^{old} = 0 \end{cases}$$

There is also the possibility to delete information described by two vectors:

$$w_{ij}^{new} = \begin{cases} 0 & \text{if } y_i x_j = 1 \\ w_{ij}^{old} & \text{if } y_i x_j = 0 \end{cases}$$

2.1.2 Retrieval

In the retrieval phase of the associative memory, a fault tolerant answering mechanism recalls the appropriate answer vector for a question vector $\vec{x}$. To the presented question vector $\vec{x}$ the most similar learned $\vec{x^l}$ question vector regarding the hamming distance is determined and the appropriate answer vector $\vec{y}$ is identified. For the retrieval rule the knowledge about the correlation of the components is sufficient, and the knowledge about the frequency of the correlation is not used. The retrieval rule for the determination of the answer vector y is:

$$y_i = \begin{cases} 1 & \sum_{j=1}^{n} \delta(w_{ij} x_j) \geq T \\ 0 & \text{otherwise.} \end{cases}$$

with

$$\delta(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \end{cases}$$

T_i is the threshold of the unit. There are many possibilities for its determination. For the two described strategies $\geq$ in retrieval equation can be replaced with $=$.

The hard threshold strategy

In the hard threshold strategy, the threshold T is set to the number of "one" components in the question vector. If one uses this strategy it is quite possible that no answer vector is determined. This happens when the question vector has a subset of components which were not correlated with the answer vector. This means that only the subset of the learned question vector is recognized, provided that the overlapping of learned patterns is not considered.

The soft threshold strategy

In this strategy, the threshold is set to the maximum sum $\sum_{j=1}^{n} \delta(w_{ij}x_j)$:

$$T :=_{max\ i} \sum_{j=1}^{n} \delta(w_{ij}x_j)$$

In this strategy, there is no answer in the case that all components of the question vector are not correlated, or, in other words, if the maximum sum is zero.

Backward projection

In this case, $\vec{y}$ is the question vector, and the answer vector which should be determined is $\vec{x^l}$. The categorization rule for the determination of the answer vector $\vec{x^l}$ is:

$$x_j^l = \begin{cases} 1 & \sum_{i=1}^{m} \delta(w_{ij}y_i) \geq T^* \\ 0 & \text{otherwise.} \end{cases}$$

This means that the synaptic matrix used is a transpose of the matrix W which is used for the forward projection.

Reliability of the Answer

Once an answer vector is determined, it would be useful to know how reliable it is [12]. Let $\vec{x}$ be the question vector and $\vec{y}$ the answer vector that was determined by the associative memory. First, the vector $\vec{x^l}$ which belongs to the vector $\vec{y}$ is determined. These two vectors form together a vector pair $\vec{x^l}\ \vec{y}$ which is stored in the associative memory. It was either created by learning, $\vec{x^l}$ and $\vec{y}$ were learned together, or created through overlap with other already learned vector pairs. The vector $\vec{x^l}$ is determined by a backward projection of the vector $\vec{y}$, with the constraint that the same threshold strategy is used as in the forward projection that determined the vector $\vec{y}$. In the second step, the similarity of the stored question vector $\vec{x^l}$ to the actually presented vector $\vec{x}$ is determined. The greater the similarity of the vector $\vec{x^l}$ to the vector $\vec{x}$, the more reliable the answer vector $\vec{y}$. In order to model cognitive processes, a normalized contrast model [213, 212, 185] is used which combines the correlation and the hamming distance between the vectors.

$$Sim(\vec{x^l}, \vec{x}) = a \sum_{j=1}^{n} x_j^l x_j - b|\vec{x^l} - \vec{x}|$$

If n is the dimension of the vectors and c the number of ones in the stored vector $\vec{x^l}$, then $a = 1/c$ and $b = 1/n$. The reliability of the answer vector $\vec{y}$ is given by $Sim(\vec{x^l}, \vec{x})$. We call $Sim(\vec{x^l}, \vec{x})$ the quality criterion of the vector $\vec{y}$,

$$qc(\vec{y}) := \frac{\sum_{j=1}^{n} x_j^l x_j}{c} - \frac{|\vec{x^l} - \vec{x}|}{n}$$

After a question vector is posed two phases follow. In the first phase, the answer vector is identified through forward projection of the associative memory. In the second phase, the backward projection determines the quality criterion, **qc** of this answer. Its value is within the range $[-1, 1]$, with -1 indicating no reliability at all and 1 absolute reliability.

2.1.3 Analysis

Storage capacity For an estimation of the asymptotic number of vector-pairs $(\vec{x}, \vec{y})$ which can be stored in an associative memory before it begins to make mistakes in retrieval phase, it is assumed that both vectors have the same dimension n. It is also assumed that both vectors are composed of M 1s, which are likely to be in any coordinate of the vector. In this case it was shown [143, 67, 189] that the optimum value for M is approximately

$$M \doteq \log_2(n/4)$$

and that approximately [143, 67]

$$L \doteq (\ln 2)(n^2/M^2)$$

of vector pairs can be stored in the associative memory. This value is much greater then n if the optimal value for M is used. In this case, the asymptotic storage capacity of the Lernmatrix model is far better than those of other associative memory models, namely 69.31%. This capacity can be reached with the use of sparse coding, which is produced when very small number of 1s is equally distributed over the coordinates of the vectors [143, 195]. For example, in the vector of the dimension n=1000000 M=18, ones should be used to code a pattern. The real storage capacity value is lower when patterns are used which are not sparse or are strongly correlated to other stored patterns.

Weight matrix diagram The diagram of the weight matrix illustrates the weight distribution which results from the distribution of the stored patterns [112, 48]. Useful associative properties result from equally distributed weights over the whole weight matrix. Clusters in the diagram indicate strong correlation between parts of stored patterns. The load of the associative memory is indicated by the percentage of weights which are not zero. A high percentage indicates an overload and the loss of its associative properties. Fig. 2.3 represents a diagram of a high loaded matrix with equally distributed weights.

Structure of weight matrix The structure of the weight matrix indicates the elementary blocks which compose an associative memory. It is represented by the frequency of different sum values of the weights of rows or columns [112]. The sum over column i is,

$$\mu_i = \sum_{j=1}^{n} \delta(w_{ij})$$

with

$$\delta(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0. \end{cases}$$

The μ_i are sorted with a new index $\pi = \iota(i)$,

$$\mu_1 \le \mu_2 \le \mu_3 \le \mu_\pi \le \ldots \le \ldots \le \mu_m.$$

The number ζ of groups with different μ values and the number of their elements is determined,

$$\underbrace{\mu_1 = \mu_2 = \mu_3}_{\tau_1=3} < \underbrace{\mu_\pi = \ldots}_{\tau_2} < \underbrace{\ldots = \mu_m.}_{\tau_\zeta}$$

This can be represented as a procedure:

```
$\Phi = 1$
$\tau_\Phi = 1$
FOR $\pi = 1$ TO m-1 STEP 1
    DO
    IF $\mu_\pi = \mu_{\pi+1}$ THEN $\tau_\Phi = \tau_\Phi + 1$
```

```
            ELSE DO $\Phi = \Phi + 1$; $\tau_\Phi = 1$ OD
    OD
$\zeta = \Phi$.
```

The ζ sorted different τ_Φ values are represented by a diagram. The x axis represents the $\Phi \in [1, 2, \ldots, \zeta]$ values and y axis the corresponding frequency of sum values τ_Φ. The relationship between the x axis ordinate and corresponding value μ_π is represented additionally, for example, by an additional plot. If the associative memory performs hetroassocative recalls, the associative matrix is not symmetric and the diagrams for the sum of rows and columns are different. The sum over row j is

$$\mu_j = \sum_{i=1}^{n} \delta(w_{ij}).$$

There are n μ_j values (see fig. 2.2). In fig. 2.4 the structure of the weight matrix of fig. 2.3 is represented. The plot illustrates that the weight matrix is composed of approximately 300 elementary blocks which represent a nearly gausian correlation between the stored pattern parts. Fig.. 2.3 shows the distribution results of the ten randomly set ones in the 2000 dimensional, 20000 learned vector pairs.

2.1.4 Implementation

The associative memory can be implemented using digital [148, 147, 150] or optical hardware [228, 85, 55, 106]. On a serial computer a pointer representation can save memory space if the weight matrix is not overloaded [14]. This fact is important when the weight matrix becomes very large. In the pointer format only the positions of the vector components unequal to zero are represented. This is done, because most synaptic weights are zero. For example, the binary vector [0 1 0 0 1 1 0] is represented as the pointer vector (2 5 6), which represents the positions of "ones". For a matrix each row is represented as a vector. The pointer matrix does not consider the information of frequency, it represents only the positions of components unequal to zero in the corresponding row.

For an unclipped weight matrix the frequencies are represented additionally by the shadow matrix. For each position in the pointer matrix, a corresponding number in the shadow matrix represents the frequency. For example, the vector [0 1 0 0 7 3 0] is represented by the pointer vector (2 5 6) as before, and by the additional shadow vector (1 7 3). This representation can still save memory space if the weight matrix is not overloaded.

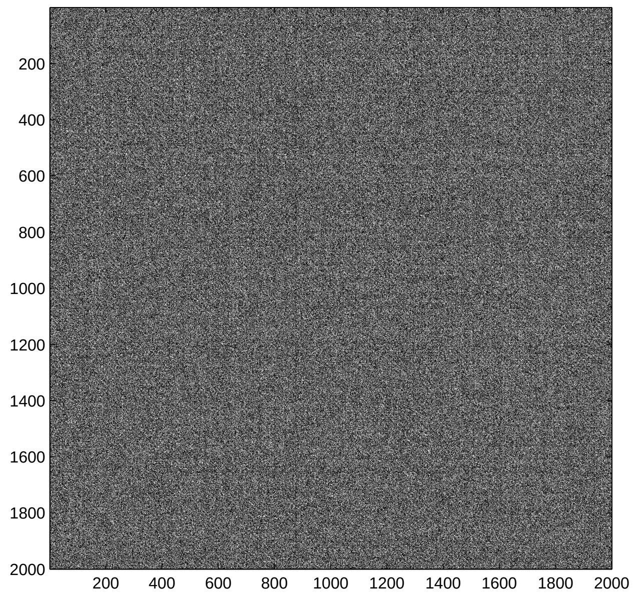


Figure 2.3: The weight matrix after learning of 20000 test patterns, in which ten ones were randomly set in a 2000 dimensional vector represents a high loaded matrix with equally distributed weights. This example shows that weight matrix diagram often contains nearly no information. Information about the weight matrix can be extracted by the structure of weight matrix, see section 2.1.3. (White color represents wights.)

An example of the representation of an unclipped weight matrix by the pointer matrix representing positions of components unequal to zero, and the shadow matrix representing for each position in the pointer matrix the corresponding frequency:

$$\underbrace{\begin{bmatrix} 2 & 0 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 4 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{weight\ matrix} \longrightarrow \underbrace{\left(\underbrace{\begin{pmatrix} 1 & \\ 1 & \\ 1 & \\ 2 & 3 \\ 2 & \\ 3 & \end{pmatrix}}_{pointer\ matrix} \ and \ \underbrace{\begin{pmatrix} 2 & \\ 1 & \\ 1 & \\ 1 & 1 \\ 4 & \\ 1 & \end{pmatrix}}_{shadow\ matrix} \right)}_{pointer\ format}$$

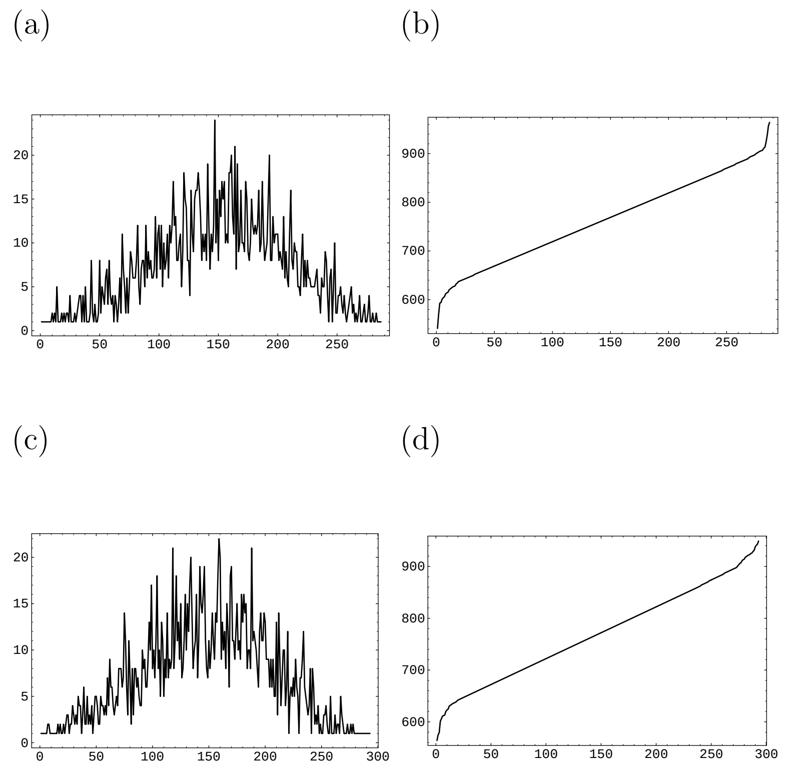


Figure 2.4: 38% of synapses of associative memory are not zero. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows. (d) The corresponding sum values of rows.

2.2 Applications

2.2.1 Word Recognition

In this application, with the aid of the associative memory, an unambiguous adress is assigned to each word with possible type errors. The ideas for the used robust mechanism come from psychology and biology [225, 226, 164, 14]. Words are represented as sequences of context-sensitive letter units. Each letter in a word is represented as a triple, which consists of the letter itself, its predecessor, and its successor. For example, the word *desert* is encoded by six context-sensitive letters, namely: *_de, des, ese, ser, ert, rt_*. The character "_" marks the word beginning and ending. Because the alphabet is composed of 26+1 characters, 27^3 different context-sensitive letters exist. In the 27^3 dimensional binary vector each position corresponds to a possible context-sensitive letter, and a word is represented by indication of the actually present context-sensitive letters. Similarly written words are represented by similarly coded binary vectors, so that type errors can be tolerated. A context-sensitive letter does not need to be a triple. It can also be a tupel consisting of the letter itself and its predecessor. Tupel representation of the context-sensitive letters is more ambiguous than the representation with the triples, but it does not require as much storage space. There are only 27^2 possible tupels. In an implementation, one must find a balance between the use of storage space and the ambiguity of the word representation. In general, a context-sensitive letter can consist of any number of letters, but only the numbers two, three (Wickelfeature) and four letters seem useful. Each word and address is represented by a unit [114, 14]. The stored binary weights represent the presence or absence of a context-sensitive letter, and the unit indicates the corresponding address. Each time a new word is entered, it is tested tp see if it was already stored. The coded vector which describes this word is posed at the corresponding positions and for each unit, the sum of the product of the corresponding ones of this vector and weights is calculated. The observer is then notified about each name whose feature sum is maximal (see fig. 2.5). In the case that the word was not recognized, it is learned through the extension of the associative memory by a new unit which represents the new word [194].

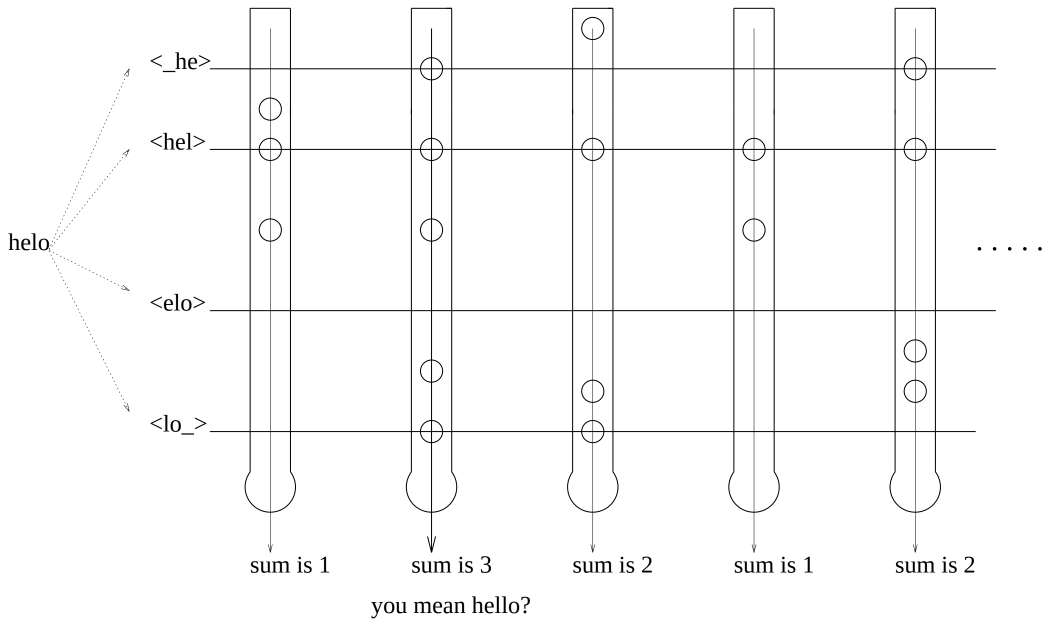


Figure 2.5: Assignment of an address to a misspelled word.

When an associative word interface (short *awi*) is integrated in bigger systems, two operation modes are needed. In the first mode, a word is entered and the address is determined. In the second, the address is given and the word that belongs to it is identified. A word could be reconstructed unambiguously with a set of given Wickelfeatures [164], but not with a set of given tupel representations. In our approach, each word is represented through context-sensitive letters and additionally as strings of characters. The first representation is used when a word is entered and an address is determined as stated earlier; the second representation is used to display a word when an address is given. In string representation [8], each symbol of the alphabet has an unequivocal number, for example as with the ASCII code. This number is represented as a one-of-N code (the position of the one represents the number). The whole string is represented as a one behind the other representation of the symbols.

2.2.2 Associative Categorization

In this application different combinations of features describe different categories. To a given question vector which describes the features, an answer vector which describes categories is associated. The presence of a feature is represented by a “one” at the corresponding position of a binary vector, its absence by a zero. Different vectors which describe the presence or absence of features correspond to different categories. The model is composed of three modules. One modul is the associative memory in which the correlation between the features and the corresponding categories is stored. The other moduls are two *awi's* for the input/output operations (see fig. 2.6). The first *awi* serves for the specification or identification of the features, the second *awi* for the specification or the indication of the correlated categories.

In the learning phase the correlated features and categories are stored. In the retrieve phase, a category or some categories are determined for a given set of present features. In the learning phase, before the vector pair which describes the features and categories can be stored, two filters must be passed. First, to ensure that the so described associations do not contradict the already stored knowledge and second, that it can be stored without errors. In the retrieve phase, after the user has specified the present features, the question vector which describes these features is posed. In addition, a resulting possible answer vector which describes a category or some categories and its quality criteria are determined with the soft threshold strategy. If an answer vector exists, it is propagated backward and the corresponding already learned question vector is determined and displayed. The following small knowledge base which identifies animals [229] and which can be easily extended was learned (see also fig. 2.7):

1. If gives milk $\wedge$ eats meat $\wedge$ has pointed teeth $\wedge$ has claws $\wedge$ has forward pointing eyes $\wedge$ has tawny color $\wedge$ has dark spots then cheetah.

2. If gives milk $\wedge$ eats meat $\wedge$ has pointed teeth $\wedge$ has claws $\wedge$ has forward pointing eyes $\wedge$ has tawny color $\wedge$ has black strips then tiger.

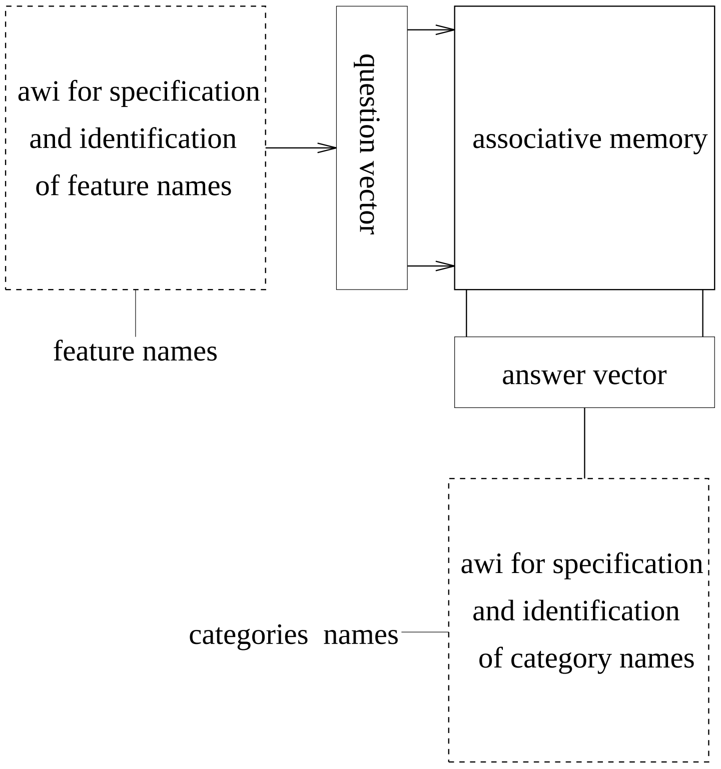


Figure 2.6: The model is composed of three modules: an associative memory in which the correlation between the features and the corresponding categories is stored, and two *awi's* for the input/output operations.

3. If gives milk $\wedge$ has dark spots $\wedge$ has hoofs $\wedge$ chews cud $\wedge$ has long legs $\wedge$ has long neck $\wedge$ has tawny color then giraffe.
4. If gives milk $\wedge$ has hoofs $\wedge$ chews cud $\wedge$ has white color $\wedge$ has black stripes then zebra.
5. If has long legs $\wedge$ has long neck $\wedge$ has feathers $\wedge$ lays eggs $\wedge$ is black and white then ostrich.
6. If has feathers $\wedge$ lays eggs $\wedge$ does not fly $\wedge$ is black and white $\wedge$ swims then penguin.
7. If has feathers $\wedge$ lays eggs $\wedge$ is a good flyer then albatros.

Knowledge Revision

For the purpose of the first filter, the question vector which describes the features which correspond to the association that should be learned are posed and

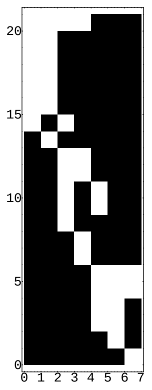


Figure 2.7: The weight matrix for the identification of seven animals.

a resulting possible answer vector is determined. If an answer vector exists, it is propagated backward and the corresponding already learned question vector is determined. Suppose that after the seven associations were stored in the associative memory, a new association should be learned: If has feathers ∧ lays eggs then robin.

```
LEARNING:

INPUT features

Nr.1 input the description >> feathers
is HAS_FEATHERS the desired description with qc=0.66 (y/n) ?  > y

|------> another fact (y/n) ?  > y
Nr.2 input the description >> lay eggs
is LAYS_EGGS the desired description with qc=1 (y/n) ?  > y
|------> another fact (y/n) ?  > n

OUTPUT categories

Nr.1 input the description >> robin
|------> another fact (y/n) ?  > n
---------------------------------------------

HAS_FEATHERS ∧ LAYS_EGGS --> ROBIN
|---> learn this rule (y/n) ?  > y
collision with the rule:
HAS_FEATHERS ∧ LAYS_EGGS --> OSTRICH ∧ PENGUIN ∧ ALBATROSS
```

The system notifies the user that the association which describes robin cannot be separated from the other association which describes the general group of birds. For the separation of this association, a feature or some features unique to robin should be entered.

For the purpose of the second filter, the vector pair which describes the association which should be learned is stored temporarily in the associative matrix. After storage, it is examined if the vectors are stored correctly, the question vector which was stored is posed, and the answer vector is determined. The two tests are now made. First, it is determined whether the answer vector which should be learned is equal to the stored answer vector. The second test determines if the quality criterion of the determined answer vector is "one". In the case that one of the answers of the two tests is negative, the user is asked for his confirmation for the storage of the association despite the fact that it cannot be learned completely correctly. If the user does not give his confirmation, the temporarily stored association is forgotten. In any other case the temporary stored association will be stored permanently. The association which should be learned: If is dangerous then tiger.

```
LEARNING:

INPUT features

Nr.1 input the description >> is dangerous
|------> another fact (y/n) ?  > n

OUTPUT categories

Nr.1
input the description >> tiger
is TIGER the desired description with qc=1 (y/n) ?  > y
|------> another fact (y/n) ?  > n
---------------------------------------------

IS_DANGEROUS --> TIGER
|---> learn this rule (y/n) ?  > y
cannot learn the information correctly with this memory.
```

This association cannot be stored without errors, because it would extend association number two which already specifies this category.

Associative Determination of Categories

In the following example associative categorization is demonstrated. The user specifies a question and two types of the toleration of errors are then performed. In the first phase syntax errors of misspelled words are tolerated and in the second phase, semantic errors. The most plausible answer to the possibly faulty or fragmentary question is then determined. This error tolerant interface and the following computation is desirable because humans are often confused and do not posses complete knowledge. The speed with which the answer is recalledis independent of the number of errors present in the specified question.

```
CONSULTING:

Nr.1 input the description >> milk
is GIVES_MILK the desired description with qc=0.39 (y/n) ?  > y

|------> another fact (y/n) ?  > y
Nr.2 input the description >> black strips
is HAS_BLACK_STRIPS the desired description with qc=0.74 (y/n) ?  > y
|------> another fact (y/n) ?  > n

Answering...

my knowledge:
GIVES_MILK ∧ EATS_MEAT ∧ HAS_POINTED_TEETH ∧ HAS_CLAWS
∧ HAS_FORWARD_POINTING_EYES ∧ HAS_TAWNY_COLOR and HAS_BLACK_STRIPS
--> TIGER with qc=1 (of the answer vector)

your input:
GIVES_MILK ∧ HAS_BLACK_STRIPS --> TIGER with qc=0.05 (of the answer vector)
```

A single category could be determined, in the second example several categories are simultaneously present,

```
CONSULTING:

Nr.1 input the description >> lays eggs
is LAYS_EGGS the desired description with qc=1 (y/n) ?  > y
|------> another fact (y/n) ?  > n

Answering...
```

```
my knowledge:
HAS_FEATHERS ∧ LAYS_EGGS --> OSTRICH ∧ PENGUIN ∧ ALBATROSS with qc=1 (of
the answer vector)

your input:
LAYS_EGGS --> OSTRICH ∧ PENGUIN ∧ ALBATROSS with qc=0.45 (of the answer
vector)
```

These simultaneously present categories correspond to the set of birds, because LAYS_EGGS is valid for all birds in the knowledge base. The "∧" indicates that the quality criterion makes a statement about the whole set.

Sparse Vectors

The more categories can be described, the smaller the ratio of the set of features which describe a category compared to the dimension of the state vector. This is because, of all possible features, only some define categories.

2.3 Conclusion

Associative categorization which relied on the associative memories was introduced. Already even in this uncomplicated application the main advantages of associative computation principles were demonstrated, namely fast computation linked with a toleration of errors. The associative categorization is an interesting alternative to the commonly used data bases techniques, because of the possibility for storing large amounts of data, and the fast associative recall.

Despite its simplicity, the associative memory model is an efficient and biologically plausible model which captures the basic behavior principles of the cell assembly theory. One associative function alone, however, is not adequate to model higher cognition. But it is, however, one of its atomic parts.

Chapter 3

Representation

Cell assemblies represent complex objects which corresponds to thoughts. Traditional symbolic forms of representation appear to be inappropriate foruse with cell assemblies because they lack a concept of similarity. In this case, categorial distributed representation of knowledge seems more suitable.

3.1 Symbolic representation

3.1.1 Logic and representation

Logical representation is motivated by philosophy and mathematics [96, 206, 111]. Predicates are functions that map objects' arguments into true or false values. They describe the relation between objects in a world which is represented by symbols (see fig. 3.1). Whenever a relation holds with respect to some objects, the corresponding predicate is true when applied to the corresponding object symbols.

Predicates can be negated by the function $\neg$ (not) and combined by the logical connectives $\vee$ (disjunction), $\wedge$ (conjunction) and the implies ($\rightarrow$) operator. $\neg$, $\vee$, $\wedge$, and $\rightarrow$ determine the predicate's value. To signal that an expression is universally true, the universal quantifier and a variable standing for possible objects is used.

$$\forall\ \mathrm{x}[\mathrm{Feathers}(\mathrm{x}) \rightarrow \mathrm{Bird}(\mathrm{x})].$$

An Object having feathers is a bird.

Some expressions are true only for some objects. This is represented by an existential quantifier and a variable.

$$\exists\ \mathrm{x}[\mathrm{Bird}(\mathrm{x})].$$

There is at least one object which is a bird.

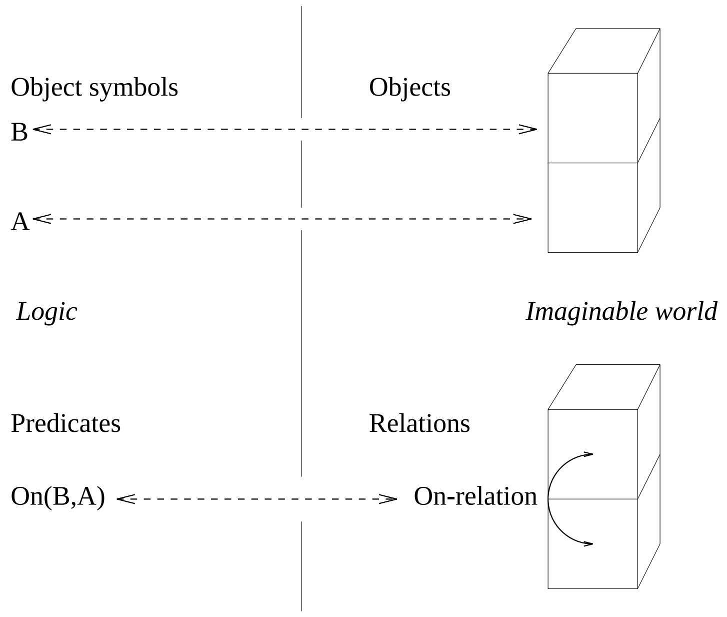


Figure 3.1: Object represented by symbols and relation represented by predicate.

An interpretation is an accounting of the correspondence between objects and object symbols and between relations and predicates. An interpretation can be only either true or false. These are some basic ideas about representation in predicate calculus, which is a subset of formal logic. A world state can be described including properties and relations using predicate calculus. This kind of description can be used to define operators like those used in the STRIPS computer science approach (see fig 3.2) [44, 136, 54].

ontable(A).
ontable(C).
on(B,A).
clear(B).
clear(C).
gripping().

3.1.2 Operators

Using the block examples, four operations "pickup", "putdown", "stack" and "unstack" can be defined [136]: [1]

[1]The expressions are always universally true, and therfore the universal quantifier is omitted.

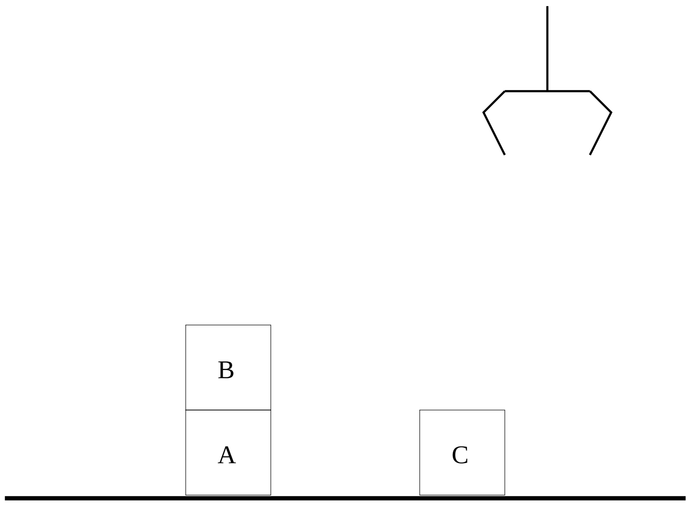


Figure 3.2: ABC block world.

$$pickup(x)\begin{cases} P: & gripping() \wedge clear(x) \wedge ontable(x) \\ A: & gripping(x) \\ D: & ontable(x) \wedge gripping() \end{cases}$$

$$putdown(x)\begin{cases} P: & gripping(x) \\ A: & ontable(x) \wedge gripping() \wedge clear(x) \\ D: & gripping(x) \end{cases}$$

$$stack(x,y)\begin{cases} P: & gripping(x) \wedge clear(x) \\ A: & on(x,y) \wedge gripping() \wedge clear(x) \\ D: & clear(y) \wedge gripping(x) \end{cases}$$

$$unstack(x,y)\begin{cases} P: & gripping() \wedge clear(x) \wedge on(x,y) \\ A: & gripping(x) \wedge clear(y) \\ D: & on(x,y) \wedge gripping() \end{cases}$$

Each of the operators is represented as triples of description. The first element is the precondition, the world state which must be met for an operator to be applied. It can be true or false when variables become identified with the values which describe the state. The second element is the additions to the state description that are a result of applying the operator. The last element is the items that are removed from the state description to create a new state when the operator is applied. These operators obey the frame axiom since they specify what is true in one state of the world and what exactly has changed by performing some action by an operator. The problem of specifying which part of the description should change and which should not is called the frame problem [229].

ontable(A).
clear(A)
ontable(C).
clear(C).
gripping(B).

The state after the operator *pickup(B)* was applied to the state of fig. 3.2, (see fig. 3.3).

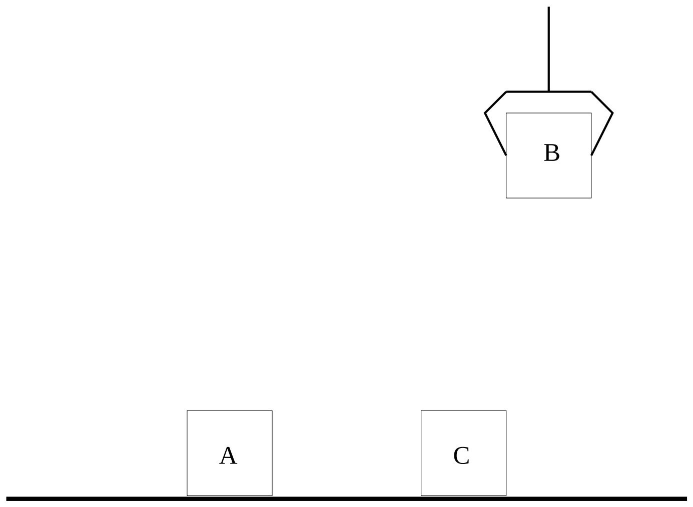


Figure 3.3: The state after the operator *pickup(B)* was applied to the state of fig. 3.2.

One problem with this kind of description of states is that no uncertainty can be expressed. For this kind of knowledge other concepts are needed.

3.1.3 Rules

A rule [229, 166, 111] contains several "if" patterns and one or more "then" patterns. A pattern in the context of rules is an individual predicate which can be negated together with arguments. The rule can establish a new assertion by the "then" part, the conclusion whenever the "if" part, the premise, is true. When variables become identified with values they are bound to these values. Whenever the variables in a pattern are replaced by values, the pattern is said to be instantiationed. Here is an example of rules with a variable x:

- $\underbrace{\text{If (flies(x)} \vee \text{feathes(x))} \wedge \text{lays eggs(x)}}_{premise}$ then $\underbrace{\text{bird(x)}}_{conclusion}$
- If bird(x) $\wedge$ swims(x) then penguin(x)
- If bird(x) $\wedge$ sings(x) then nightinagle(x)

The following assertions are present:

- feathers(Pit)
- lays eggs(Pit)
- swims(Pit)
- flies(Airbus)

Pit is a bird because the premise of the first rule is true when x is bound to Pit. Because bird(Pit), the premise of the second rule is true and Pit is a penguin.

A commonly used approach to express uncertainty is the certainty theory [175, 76]. It is motivated by the fact that the knowledge content of the rule is much more important than the algebra which holds the system together. Certainty theory splits the confidence into a confidence for and a confidence against a hypothesis. $MB(H|E)$ is the measure of a belief of a hypothesis H given the evidence E from the interval [0,1]. $MD(H|E)$ measures the disbelief of a hypothesis H given the evidence E from the interval [0,1]. The certainty factor is computed as:

$$CF(H|E) = MB(H|E) - MD(H|E) \quad \in \; [-1,1]$$

The evidence for a hypothesis is stronger as CF approaches 1 and grows stronger against a hypothesis as the CF approaches -1. CF values are determined by an observer who specifies the rules. They are attached to the "if" patterns and to the "then" patterns. The CF of the patterns are combined in the following manner:

$$CF(P2 \wedge P2) = MIN(CF(P1), CF(P2))$$

$$CF(P2 \vee P2) = MAX(CF(P1), CF(P2)$$

The combined CF of the "if" pattern is then multiplied by the CF of the "then" patterns. If, for example, P1 has CF of 0.6, P2 CF of 0.4, and P3 CF of 0.2 and the rule is:

(P1 $\vee$ P2) $\wedge$ P3 then R1

with CF 0.7 of R1, then:

$$MIN(MAX(0.6, 0.4), 0.2) \cdot 0.7 = 0.14$$

When two or more rules support the same result, the CF of these results are combined together. Suppose $CF(R1)$ and $CF(R2)$ are the present certainty factors associated with the result R, so the new CF of the result is calculated as:

$$\begin{array}{lc} CF(R1) + CF(R2) - (CF(R1) \cdot CF(R2)) & if\ CF(R1) > 0\ and\ CF(R2) > 0 \\ CF(R1) + CF(R2) + (CF(R1) \cdot CF(R2)) & if\ CF(R1) < 0\ and\ CF(R2) < 0 \\ \dfrac{CF(R1) + CF(R2)}{1 - MIN(|CF(R1)|, |CF(R2)|)} & otherwise. \end{array}$$

3.1.4 Frames

Frames describe individual objects and entire classes [123, 124, 229]. They are composed of slots which can be either attributes which describe the classes or object, or links to other frames. With the aid of links, a hierarchy can be represented in which classes or objects are parts of more general classes. In this taxonomic representation, frames inherit attributes of the more general classes (see fig. 3.4). Frames can be viewed as generalization of semantic nets. They are psychologically motivated and were popularized in computer science by Marvin Minsky. One important result of the frame theory is the object oriented approach in programming.

3.1.5 Similarity

Similarity in symbolical systems is defined by the relations between the represented objects. These relations are defined by the whole system and not by the objects alone [125]. An example is the similarity which is defined by the distance between the objects in a taxonomy. Because of the global nature of this similarity, it is difficult to define a corresponding vector space which allows the representation of those objects by vectors.

3.2 Categorial representation

Humans divide the world into categories so that they can make sense of it [185]. The categorization task consists of the determination if an object belongs to a category [142, 100]. The symbols based representation uses additional concepts to represent uncertainty or heuristics. Categorical representation uses one concept [184]. There are two different classes of categorization tasks: the visual and the verbal categorization [185].

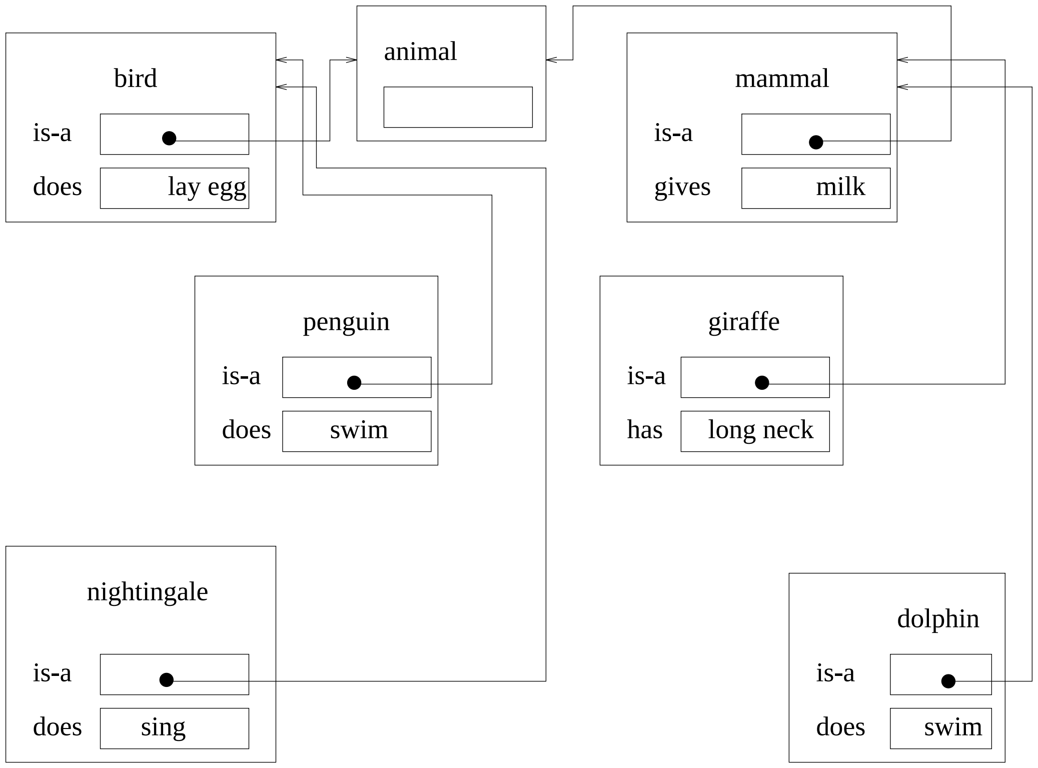


Figure 3.4: Taxonomic frame representation of some animals.

3.3 Verbal categories

3.3.1 Feature Approach

Objects can be described by a set of discrete features, such as red, round and sweet [212, 115]. The similarity between them can be defined as a function of the features they have in common [142, 198, 56, 53]. The contrast model of Tversky [212] is one well known model in cognitive psychology [185, 140] which describes the similarity between two objects which are described by their features. Suppose we have an object A and an object B, so the similarity between these object is given by:

$$Sim(A, B) = \alpha\zeta(A \cap B) - \beta\zeta(A - B) - \gamma\zeta(A - B)$$

$(A \cap B)$ gives the common features between A and B. $(A - B)$ designates the set of features distinct to A and $(B - A)$ the set of features distinct to B. The function ζ describes the salience of the features, and α, β, γ are the constants that define the relative contribution of each of the three feature sets. The asymmetry of the similarity function is one of the psychological demands [212, 57, 185]. Psychological experiments suggest that unfamiliar categories are judged more equal to familiar then the other way around. For example, pomegranate is judged more similar to apple by most people than apple is to pomegranate [185]. This demand can be reached by the contrast model if β exceeds γ. This is because the familiar categories are described by more features then the unfamiliar ones. For simplicity we assume that all features are equally salient; the function ζ simply assigns a value 1 to each feature.

apple	pomegranate
red	red
round	round
hard	
sweet	
trees	

Table 3.1: Apple and pomegranate description by features.

$$Sim(pomegranate, apple) = \alpha(2) - \beta(0) - \gamma(3)$$

$$Sim(apple, pomegranate) = \alpha(2) - \beta(3) - \gamma(0)$$

If it is supposed that the similarity value should be from the interval [-1,1], with -1 indicating no similarity at all, the normalized contrast model is used: [2]

$$Sim(A, B) = \alpha|A \cap B| - \beta|A - B| \quad \in (-1, 1]$$

$|A|$ is the number of the present features of set A and $|A \cup B|$ the number of different features of the two sets A and B, so $\alpha = 1/|A|$ and $\beta = 1/|A \cup B|$.

$$Sim(pomegranate, apple) = 2/2 - 3/5 = 4/5 = 0.8$$

$$Sim(apple, pomegranate) = 2/5 - 3/5 = -1/5 = -0.2$$

Objects can be represented by binary vectors as was done previously in chapter 2. A feature is represented by a "one" at the corresponding position of a binary vector, its absence by a zero. In this case, n is the dimension of the vectors and c the number of the "ones" in the vector $\vec{a}$.

$$qc_a(\vec{b}) := Sim(\vec{a}, \vec{b}) = \frac{\sum_{j=1}^{n} a_j b_j}{c} - \frac{|\vec{a} - \vec{b}|}{n}$$

This is the quality criterion of the object b in regard to object a, with $qc_a = 1$ indicating absolute similarity and $qc_a \to -1$ no similarity at all.

[2]However Sim(A,B) can never reach the value -1 because $|A - B| \neq |A \cup B|$.

Categorization

An object is judged to belong to a verbal category to the extent that its features are predicted by the verbal category [141]. Since only the sets of prototypical features which define the categories are considered during verbal categorization of some categories to a feature set, the normalized contrast model can be reduced. If Ca is a category and B the features set, so only $Ca \cap B$ features are considered:

$$Simc(Ca, B) = \frac{|Ca \cap B|}{|Ca|} - \frac{|(Ca - (Ca \cap B)|}{|Ca|} = \frac{2}{|Ca|} \cdot |Ca \cap B| - 1 \quad \in [-1, 1]$$

$|Ca|$ is the number of the prototypical features that define the category Ca. The present features are counted and normalized so that the value can be compared. For example, the category **bird** is defined by the following features: flies, sings, lays eggs, nests in trees, eats insects. The category **bat** is defined by the following features: flies, gives milk, eat insects. The following features are present: flies and gives milk.

$$Simc(\mathbf{bird}, presentfeatures) = \frac{1}{5} - \frac{4}{5} = \frac{2}{5} \cdot 1 - 1 = -\frac{3}{5}$$
$$Simc(\mathbf{bat}, presentfeatures) = \frac{2}{3} - \frac{1}{3} = \frac{2}{3} \cdot 2 - 1 = \frac{1}{3}$$

In binary vector notation c is the number of the ones in the vector $\vec{C}$ which describes the category Ca, and $\vec{f}$ the vector which describes the present features.

$$qc^f(Ca) := Simc(\vec{C}, \vec{f}) = \frac{2}{c} \cdot \sum_{j=1}^{n} C_j f_j - 1$$

This is the quality criterion of the category Ca for the given feature set f represents the rating in the presence of a category, with $qc^f(Ca) = 1$ an absolute rating and $qc^f(Ca) = -1$ no rating at all.

3.3.2 Uncertainty and salience

Salience of a Feature Features that discriminate among relevant facts should have a higher salience than those that do not [185]. The features of an equal salience have a unary representation, they can only be represented as existent or nonexistent. A category that is described as a set of features can be present with different grades of vagueness corresponding to the cardinal number of the set. A set of features that describes a category can be sometimes divided into subsets that represent some subcategories. Each feature can be also regarded as a kind of subcategory. If this subcategory can not be described by other features, but, nevertheless, should have the properties of variable salience and vagueness, it is described by invisible features. To each feature a number of invisible features is assigned dependent on its salience.

Uncertainty of a Category An observer determines the presence of features corresponding to each category. However, if it is not possible to observe some features, the quality criterion can never reach the maximal value 1. If some features exist for a complete definition of a category, but because of the lack of available knowledge they cannot be named, they cannot be verified. These features are called the unobservable features. The certainty of the definition of a category is expressed by their number.

The number of invisible features for each feature and the number of unobservable features for each category are determined by an observer who specifies the category.

Example An example of two old sayings from country folklore:

1. If it is April and it snows much then probably the apple harvest will be bad.
2. If it is April and it rains a lot and it is very cold then the vintage will be good.

The number of invisible features as determined by the observer:

- **April** is represented by one invisible feature, as it can be present or absent.
- **snows much** is described by two invisible features because the observer thinks that it has a higher salience than **April**. It can be either present, maybe present, or absent.
- The observer thinks that **rains a lot** has the same salience as **snows much**.
- The observer thinks that **very cold** has the greatest salience, as it is described by three invisible features. It can be either present, maybe present, maybe absent or absent.

The number of unobservable features as determined by the observer:

- The observer thinks that the uncertainty of the first category which corresponds to the adjective *probably* is expressed by two unobservable features.

3.3.3 Representation by Associative Memory

The set of features can be represented by a binary vector in which the positions represent different features. For each category a binary vector can be defined. A "one" at a certain position corresponds to a certain invisible or unobservable feature. Those vectors can be stored in an associative memory. The two country

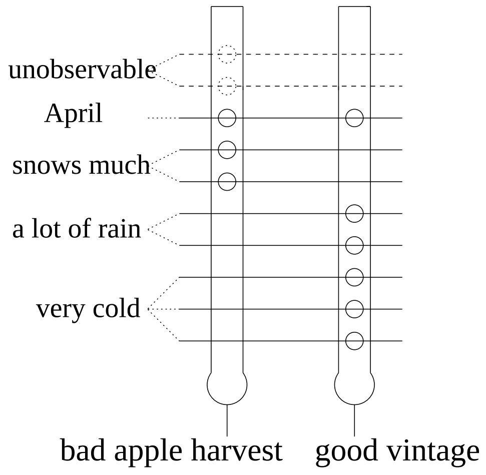


Figure 3.5: Neural representation of two country sayings

sayings are represented by an associative memory composed of two units (see Fig. 3.5).

The learning and the forgetting rules stay the same. Because the determined answer vector is composed of qc^f for each category, the retrieval rule changes to:

$$qc(Ca_i) = y_i = \frac{2}{\sum_{j=1}^{n} \delta(w_{ij})} \cdot \sum_{j=1}^{n} \delta(w_{ij} x_j) - 1$$

During the retrieval phase, the answer vector is recalled by the question vector $\vec{x}$ which represents the present feature set and the belief in their presence. The belief in the presence of a feature is represented by the observer by the number of the invisible features. Only the presence of invisible features which belong to a feature is respected, and not their position in this set. The following situation is coded by an observer and represented by $\vec{x}$ (see tab. 3.2): It is April, it rains a lot, it snows a little, and most of the time it is cold.

$$qc(bad\ apple\ harvest) = -\frac{1}{5}, \quad qc(good\ vintage) = \frac{1}{3}$$

3.3.4 Table of belief values

Salience of a feature is represented by the number of invisible features determined by an observer who specifies the category.

Belief in the presence of a feature is represented by the number of present invisible features determined by an observer who utilizes the category.

To facilitate the specification of the belief value in the presence of a feature a

feature name	$\vec{x}$
unobservable features	0
	0
April	1
snows much	1
	0
a lot of rain	1
	1
very cold	1
	0
	0

Table 3.2: The unobservable features are always zero. April's invisible feature is present. One invisible feature indicating "snows much" is present because it snows a little. Two invisible features incidcating "lot of rain" are present. One invisible feature indicating "very cold" is present as it is cold most of the times, but not very cold.

belief table is used in which different belief values are named by adjectives. The observer describes the present features with the corresponding adjectives from the table. Different tables can be specified depending on the observer and the task (tab.3.3, or tab.3.4). A chosen belief value from the table is converted to the corresponding belief value of the feature by

$$belief = \lfloor salience \cdot \frac{table\ belief}{table\ salience} \rfloor .$$

	table belief value
present	2
maybe	1
absent	0

Table 3.3: The observer who utilizes the category can chose between three different belief values. The table salience is two.

Nothing is known about unspecified features. The probability of their presence is fifty percent or less depending on their salience,

$$belief\ in\ the\ not\ specified = \lfloor \frac{salience}{2} \rfloor .$$

	table belief value
certainly	6
very probably	5
probably	4
maybe	3
probably not	2
very probably not	1
certainly not	0

Table 3.4: The observer who utilizes the category can chose between seven different belief values. The table salience is six.

3.3.5 Example

A small knowledge base of four rules for the determination of disorders of an old car [118] is represented by an associative memory (see fig. 3.6).

1. If not turn over(4) $\wedge$ lights weak(2) $\wedge$ radio weak(2) then battery bad
2. If turn over(1) $\wedge$ smell gas(4) then with [0.67] carburetor is flooded
3. If turn over(1) $\wedge$ gas gauge low(2) then with [0.20] fuel pump bad
4. If turn over(1) $\wedge$ gas gauge empty(3) then out of gas

The number of the invisible features for each feature stands in the round brackets. The number of the unobservable features if present can be determined from the maximum quality criterion value $qc_{max}(Ca)$ which the category can reach and which is represented in the square brackets by [3] :

$$unobservable\ features = \frac{1 - qc_{max}}{1 + qc_{max}} \cdot invisible\ features.$$

This formula is infered from the computation of qc_{max},

$$n = unobservable\ features + invisible\ features,$$

$$qc_{max} = \frac{invisible\ features}{n} - \frac{unobservable\ features}{n}.$$

Rule two has $1 = round((1 - 0.67)/(1 + 0.67) \cdot 5)$ unobservable features and rule three $2 = round((1 - 0.2)/(1 + 0.2) \cdot 3)$.

[3]Because qc_{max} is not given exactly, the result should be rounded.

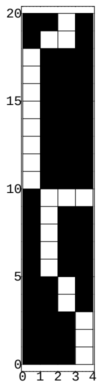


Figure 3.6: The weight matrix for car diagnosis.

The observer specifies the following features using the belief table 3.4:

1. certainly **not turn over**
2. very probably **smell gas**
3. certainly not **gauge empty**

feature name	belief
not turn over	4
lights weak	*1*
radio weak	*1*
turn over	*0*
smell gas	3
gas gauge low	*1*
gas gauge empty	0

Table 3.5: Belief values.

The quality criteria of the categories are determined by the associative memory using the determined number of invisible features (see tab.3.5), which define the question vector $\vec{x}$.

$$qc(battery\ bad) = 0.5,\ qc(carburetor\ is\ flooded) = 0,\ qc(fuel\ pump\ bad) = -0.6,$$
$$qc(out\ of\ gas) = -1$$

3.4 Visual Categories

3.4.1 Shape similarity

Psychologists have found out that, rather than relying on prototypical features, picture categorization often relies on detailed shape representation [183, 130, 185]. Objects and scenes can be represented by binary pictures which are normalized for size and orientation [42]. Similarity between the objects or scenes is measured by the amount of shared area between the overlaid patterns [15, 94, 184]. Suppose we have an object A represented by binary pattern vector $\vec{a}$ and an object B by the binary vector $\vec{b}$, so the similarity is given by:

$$Sim(\vec{a},\vec{b}) = \frac{\sum_{j=1}^{n} a_j b_j}{c} - \frac{|\vec{a}-\vec{b}|}{n} \quad \in (-1,1],$$

where n is the dimension of the vectors and c the number of ones in the vector $\vec{a}$. The shape similarity to an object is measured for a category represented by a picture. For example, the category **apple** is represented by a picture. The following binary pictures of the object are present: pear and flower [4].

$$Sim(\vec{apple},\vec{pear}) = \frac{271}{467} - \frac{85}{4096}$$

$$Sim(\vec{apple},\vec{flower}) = \frac{44}{467} - \frac{212}{4096}$$

$$Sim(\mathbf{apple}, pear) = 0.56$$

$$Sim(\mathbf{apple}, flower) = 0.04$$

The quality criterion of the object b in regard to category Ca represented as $\vec{C}$ is defined as

$$qc_{Ca}(b) := Sim(Ca,b) = \frac{\sum_{j=1}^{n} C_j b_j}{c} - \frac{|\vec{C}-\vec{b}|}{n} \quad \in (-1,1],$$

where n is the dimension of the vectors and c the number of ones in the vector $\vec{C}$ describing the category Ca.

$$qc_{apple}(pear) = 0.56,$$

$$qc_{apple}(flower) = 0.04.$$

[4]See next pages. Binary pictures of the dimension $64 \cdot 64$.

```
                       *  ** **
                    **        ****
                   *        * *  **
                   *         *****
                  *        * *
                  *      *
                  *      *
                   *     *
                   *     *    ******
                   *     *  * ***   ** **
                   *    **   *          **
                   *    *   **           *
                    *   ****
                    *  ***
                     * ***
                      ***
                      ***
                       *
                      **
                      **
          * ********  **      *******
       **** * *** *******   ************
      **            *********** *** ******
    ***                 * * *  * *** ******
   **                        * *  *** *******
   *                                  * ******
  **                                     *****
  *                                      ******
  *                                      *******
  *                                      ***** *
 *                                       *******
*                                         *  ****
*                                          ******
*                                            *****
*                                            * ***
*                                             *  *
*                                            *****
 *                                           *** *
 *                                            * **
 *                                           *****
 *                                            ****
**                                           ** **
*                                            ** **
*                                            *****
*                                            ** **
*                                            ****
**                                         * ***
 *                                         * ***
 **                                       ** ***
  *                                      * * **
  *                                       **** *
  **                                     ******
   **                                   *******
    **                                  *** **
     **                                 * ***
      *                                 *****
      **                                *****
       *                              ******
       **                        ** * ****
        **                       * *******
         ****                    ***** **
            **** **      *********  ***
           *********  *****  *** *****
                *  ****   *******
```

```
                              *** ****
                           ********   ***
                         ****          ***
                         **             ***
                     * * *****           **
                   **      * *            *
                    *         *           **
                  **       ** *
                            ***
                 *           ***
                 *            **
                 *            ***
                *             * **
                *             ****
                *               **
                                 ***
               *                 ****
                                    **
              *                     **
              *                      **
            *                         ***
           *                            **
          *                            ****
          *                              ***
         *                               ****
         *                               *****
                                          ** *
                                           ***
       *                                   ****
      **                                   *****
     *                                     * ***
     *                                      ****
    *                                        ***
   **                                       * ***
   *                                        *** *
   *                                        *****
   *                                        *** *
  **                                         * **
  *                                         *****
  *                                          ****
 **                                         ** **
 **                                         ** **
                                            *****
*                                           ** **
**                                          ****
**                                         * ***
 *                                         * ***
 **                                       ** ***
  **                                     * * **
  *                                       **** *
  **                                     ******
   **                                   *******
    **                                  *** **
     **                                 * ***
      *                                 *****
      **                                *****
       *                              ******
       **                       ** * ****
        **                      * *******
         ****                   ***** **
            **** **      *********  ***
           *********  *****  *** *****
                *  ****   *******
```

```
                      *****
                     **   *
                     *   **
                     *   *   *****
                    **   *****  **
                    **    **   **
                    **    *     *
               *******    *    **
              **   ********   **
              *     ***** ** **
               *     **   ********
               *     *     **     **
               ***   ***  ***      *
                 ********** ********
                   **   *     *
                 ** *   ***    *
    ****         *  *   * **   *
 ***   **        *  *   *  *****
 *      **      **  *   *
**       *      **   ***
**        *     *
 **       *   **
  *       *   *
   ****   * **
     **   **
      ***  *
         **
        ***
        ***
         * *   ***
         *******  *
         ** ***   **
        ** **     *
       **    **   *
      *       **  *
      ***      *  *
      **       ** *
      *         **
      *          *
      **
      *
      *
      *
      *
      *
      *
```

3.4.2 State representation

The category "tower" is represented in the blockworld as shown in fig.3.7. Different states are represented in fig. 3.8.

$$qc_{tower}(state_1) = 0.81,\ qc_{tower}(state_2) = 0.7,\ qc_{tower}(state_3) = 0.7$$

The category "tower" is most similar to the $state_1$. The qc_{tower} represents a function that rates the value of different states according to how similar a category is to them. If it is supposed that this similarity corresponds to the distance of the category from the states in a search space [214, 166], then qc_{tower} is also a heuristic function.

Transition between states can be represented as associations: two binary pictures describing the state before the transition and after the transition. This

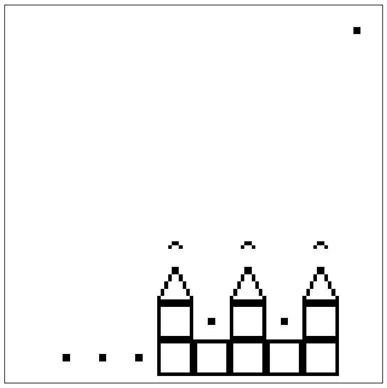


Figure 3.7: “Tower” in the blockworld.

approach requires for each possible transition a picture pair which describes the transition. The transitions must have regard to the frame problem. There are two main disadvantages of this basic approach:

- Dependent on the task, all possible associations must be determined. For example about “9!” in the 9-puzzle task [214, 111].
- When several possible associations arise from a single input they can not be learned by an associative memory [7]. Either a nonlinear mechanism is required to select one or the other branch to avoid the sum of output branches, or the output pattern must be divided in subpatterns which represent the branches.

This rigidness can be escaped [78] by the breaking of the state vector into meaningful pieces [7]. Pieces which represent objects of the scene are called cognitive entities. Cognitive entities allow the representation of any desired correlation between the objects and the representation of the total, partial, and focalized associations [78] (see fig. 3.9).

3.4.3 “What” and “Where”

Gross and Mishkkin(1977) [60] suggest that the brain includes two mechanism for visual categorization: one for the representation of the object and the other for the representation of the localization [108, 88, 155]. The first mechanism is called the “what” pathway and is composed of the temporal lobe. The second mechanism is called the “where” pathway and is composed by the partial lobe [88, 155]. According to this division, the identity of a visual object can be coded apart from the location and the size of the object. A visual state can be represented by cognitive entities. Each cognitive entity represents the identity of the object and its position by the coordinates. The identity of an object is represented

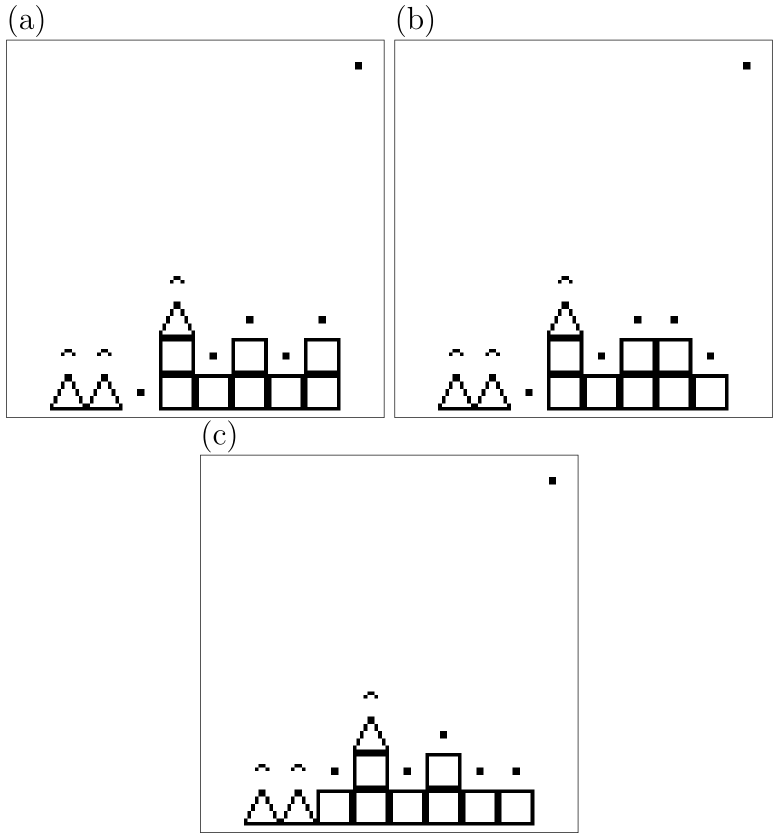


Figure 3.8: (a) $state_1$. (b) $state_2$. (c) $state_3$.

by a binary pattern which is normalized for size and orientation. Its location corresponding to the abscissa is represented by a binary vector of the dimension of the abscissa of the picture representing the state. The location corresponding to the ordinate is likewise represented by a binary vector of the dimension of the ordinate of the picture representing the state. A binary bar of the size and position of the object in the picture of the state represents in each of those vectors the location and size (see fig. 3.10, fig. 3.11 and fig. 3.12).

The three vectors from which the cognitive entity is composed are called associative fields. A scene can be represented either by a picture or an equivalent by cognitive entities which can be represented by a unit (see fig. 3.13).

3.4.4 Associations

Cognitive entities can represent associations which represent transitions between states. The first pattern represented by the cognitive entities describes the state which should be present before the transition (the premise). The second pattern describes the world state after the transition (the conclusion). In order to preserve

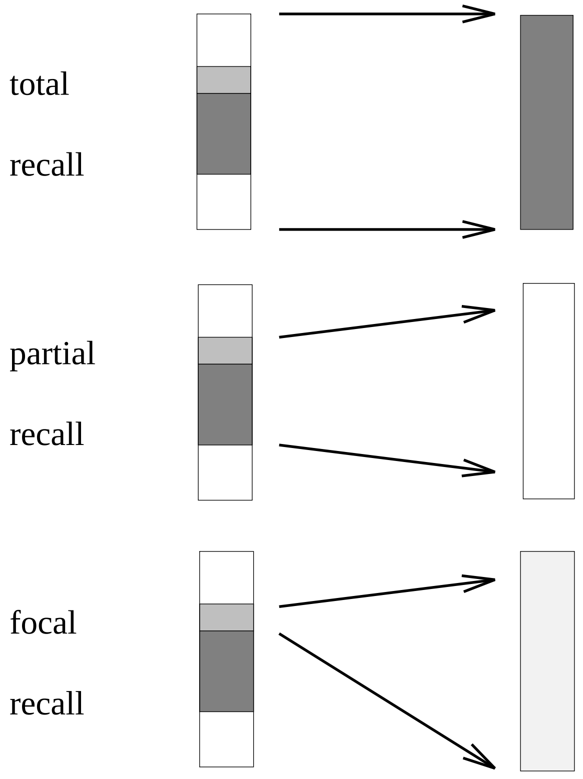


Figure 3.9: Total, partial, and focalized assocations.

the equality of cognitive entities in the premise and in the conclusion pattern, a notation for an empty cognitive entity is used (see fig 3.14). The associations are defined with respect to the frame problem. In fig. 3.14 an example from the block world is shown. Both, an empty roboter arm, which is represented by the right corner, or a "clear" position are represented by a dot.

The cognitive entities of the premise pattern are replaced by the conclusion pattern in case the similarity between the condition pattern and the corresponding part of the state picture is sufficient,

$$Sim(condition\ pattern, state\ pattern) > threshold$$

The accepted uncertainty is dependent on the threshold value.

3.5 Conclusion

Categories were shown to be the basic building blocks of knowledge representation when human problem solving is simulated by neural networks. Distributed representation and similarity are the natural properties of categorical representation and it is this properties which distinguishes categorical representation from

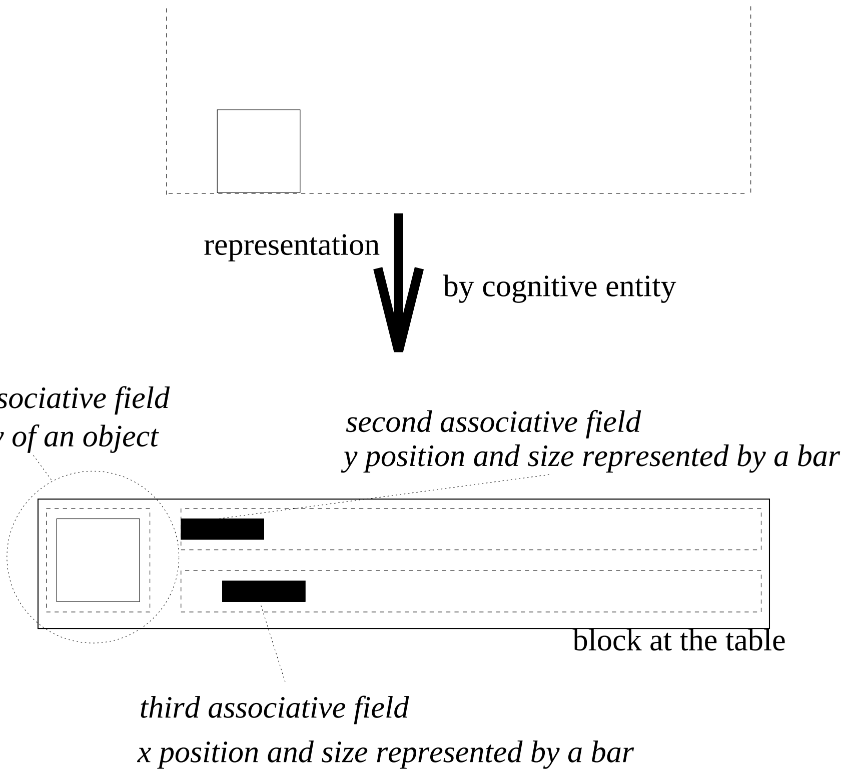


Figure 3.10: A cognitive entity.

symbolical representation. Similarity is used when uncertain knowledge is represented, without the need of an additional calculus. In addition belief tables allow the detachment of the uncertainty of the coded knowledge and of the actually present knowledge. A heuristic function results from the similarity principle without the need for any additional definition. It can speed up the search during problem solving in the problem space.

The introduced vectorial representation is as an alternative to symbolical representation. The similarities are described by the quality criterion.

3.5.1 Quality criterion

Different quality criteria were defined as a gage in the binary vectorial representation:

- $qc(\vec{y})$, quality of the answer $\vec{y}$ of an associative memory.
- $qc^f(Ca)$, quality of a category for a feature set f.
- $qc_{Ca}(b)$, quality of an object b in regard to category Cs.

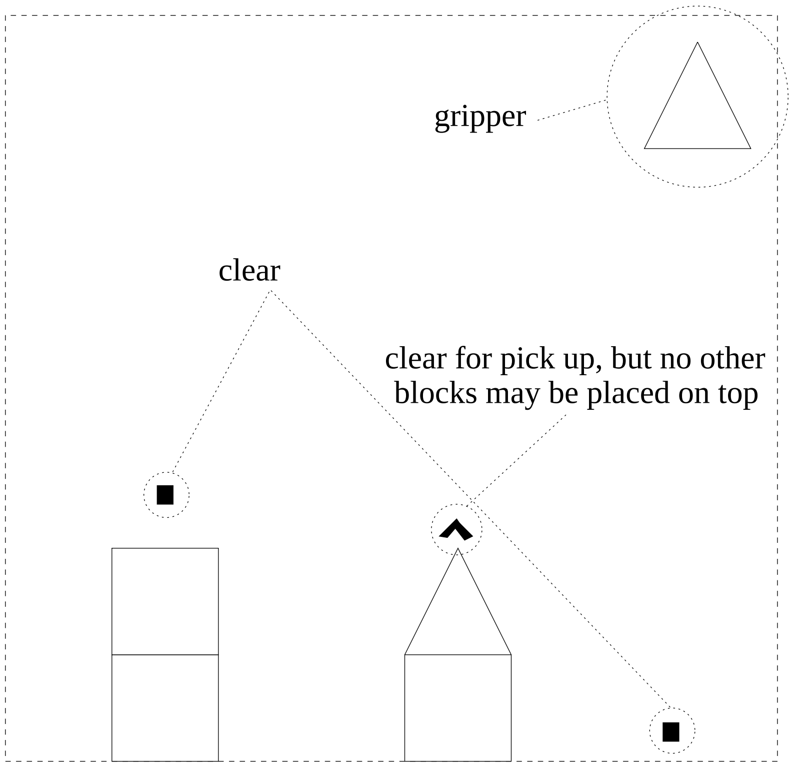


Figure 3.11: A state in the blocks world.

Because the search for the best similarity measure between binary vectors dependent on the task is not the focus of this work, a psychologically motivated measure was defined which uses both correlation and the hamming distance. Through normalization, quality values of different answers can be compared.

3.5.2 Verbal categorization

Verbal categories were represented by features of different salience. The quality criterion represents a rating in the presence of a category which is based on counting. The representation of categories by an associative memory and the belief table offer an easy access to uncertain knowledge.

3.5.3 Visual representation

Principles of pictorial categorization were illustrated. These principles were used for the state representation of the world. The biologically and physiologically "what" and "where" motivated representation of pictures through cognitive entites offer an efficient way for the representation of associations which can handle uncertainty by the similarity principle.

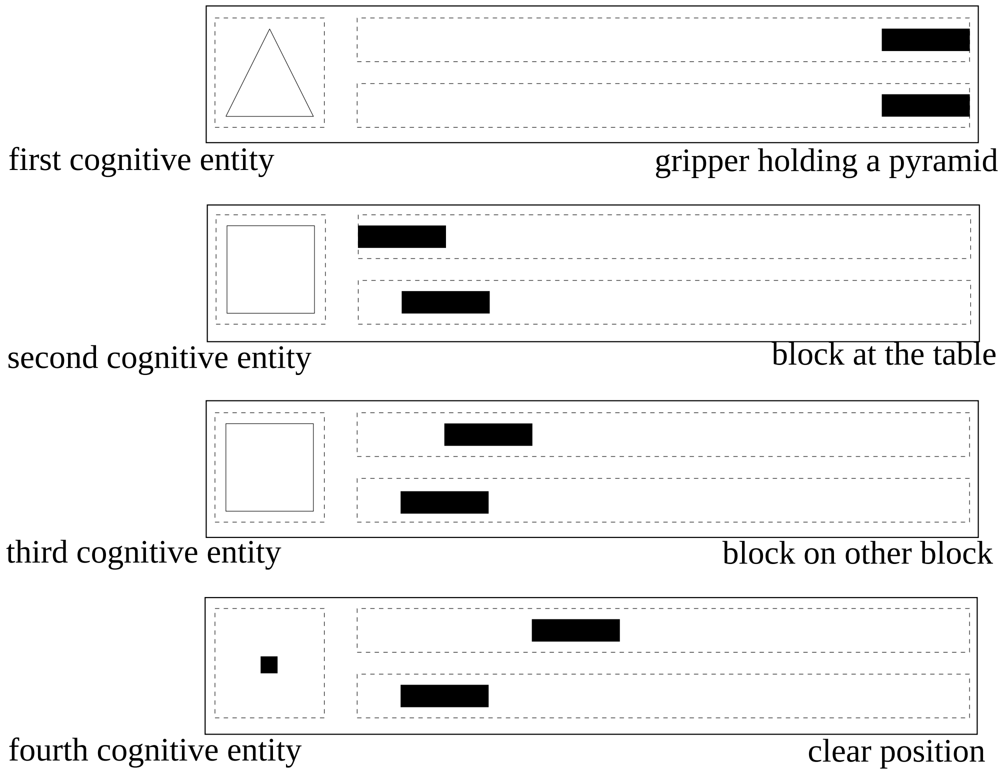


Figure 3.12: Part of the blocks world state (see fig. 3.11) represented by four cognitive entities. For the representation of the whole state four additional cognitive entities are needed.

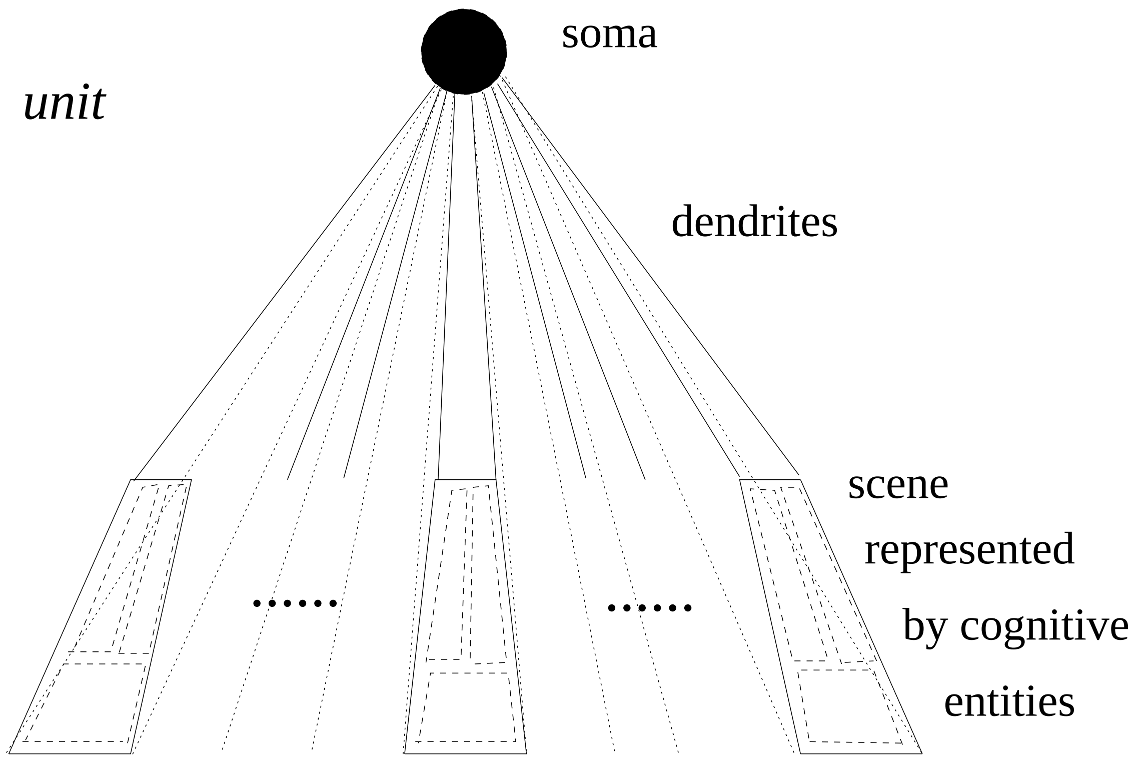


Figure 3.13: A scene represented by a unit.

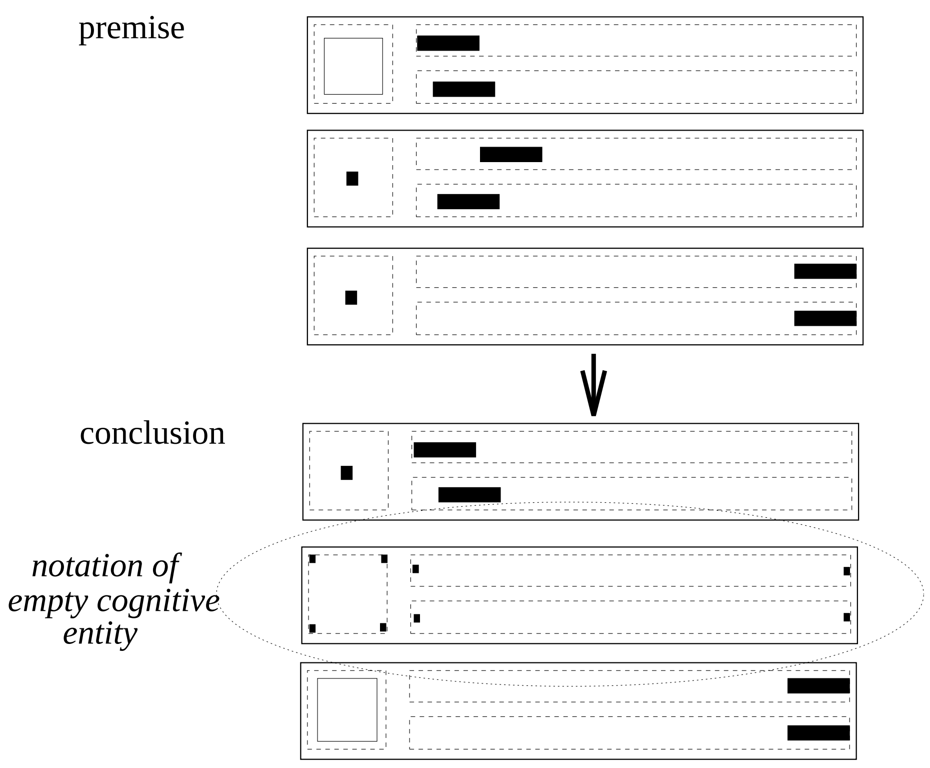


Figure 3.14: Representation of the association: If a block is at a certain position and above it it is clear and the gripper is empty then the block is grasped by the gripper. Because of the frame problem, the old position of the block is represented as clear. One cognitive entity of the conclusion pattern is not used. In the inverse association, the premise pattern is interchanged with the conclusion pattern.

Chapter 4

Hierarchical Categorization

4.1 Production Systems

Problem-solving can be modeled by a production system which implements a search algorithm. It is closely related to the approach taken by markov algorithms [113], and, like them, it is equivalent in power to a Turing machine [211]. The production system is also a model of actual human problem-solving behavior [134, 6, 84, 133]. It is composed of [27, 111]:

- Set of rules. These rule are also called productions. The set of rules models the human long term memory.
- Working memory. This memory contains a description of the state in a problem solving process. The state is described using predicate calculus and is simply called a pattern. Whenever a premise is true, the conclusions of the rules changes the contents of the working memory. The working memory models the human short term memory.
- Recognize-act cycle. The current state of the problem-solving process is maintained as a set of patterns in the working memory. Working memory is initialized with the initial state description. The patterns in working memory are matched against the premise of the rules. The premise of the rules which match the patterns in working memory produces a set which is called the conflict set. One of the rules of this set is chosen and the conclusion of the rule changes the content of the working memory. This process is denoted as firing of the rule. This recognize-cycle is repeated on the modified working memory until a desired state is reached or no rules can be fired. The recognize-act cycle models the current focus of attention triggering one of the set of permanent skills. This, in turn, changes the focus of attention.

Conflict resolution chooses a rule from the conflict set for firing. There are different conflict resolution strategies, such as choosing a random rule from the set, or selecting a rule by some certain function. In a pure production system which was proposed as a formal theory of computation (Post 1943) [157] the system halts if no production can fire in a state. In systems which model human behavior and in practical applications, backtracking to a previous state of working memory is allowed. By allowing backtracking and the exclusion of loops, a search from the initial state to the desired state is executed. The search defines a problem space and can be represented as a tree. However, it may not reach the desired goal either because the branches are infinite, or because after backtracking to the initial state no rule can fire.

Human problem-solving can be described by a problem-behavior graph constructed from a protocol of the person talking aloud, mentioning moves considered and aspects of the situation. According to the resulting theory, problems are solved by searching in a problem space whose state includes the initial situation and the desired situation [134, 133, 8]. The short term memory is limited to holding the temporary state. Progressive depending and local focusing leads to a depth-first search [133].

4.1.1 Heuristic function

A heuristic function is used that rates the value's different states according to how far they are from the desired state. The best-first search is the search where the best rule (according to a heuristic function of the conflict set) is chosen. The better the heuristic measure of the remaining distance to the desired state, the faster the best-first search [229].

4.2 Deduction systems

Problems without side effects of actions can be described by deduction systems which are a subgroup of production systems [229]. In deduction systems the premise specifies combinations of assertions, by which a new assertion of the conclusion is directly deduced. This new assertion is added to the working memory. Deduction systems do not need strategies for conflict resolution because every rule presumably produces reasonable assertions and there is no harm in firing all triggered rules. Deduction systems may chain together rules in a forward direction, from assertions to conclusions, or backward from hypotheses to premises. During backward chaining it is ensured that all features are properly focused. Backward chaining is used if no features are present. If all features are given, forward chaining is used to prevent wasting of time pursuing hypotheses which are not specified by the features. The chained rules describe the complete problem space which can be represented by a semantic net [159, 172]. For clarity, rules can be arranged in groups [3, 82] which define a taxonomy according to their dependence between the conclusion pattern and one premise pattern (see fig. 4.1).

begin of the forward chaining

- If (flies(x) $\vee$ feathes(x)) $\wedge$ lays eggs(x) then bird(x)
- If gives milk(x) then mammal(x)

bird group

- If bird(x) $\wedge$ swims(x) then penguin(x)
- If bird(x) $\wedge$ sings(x) then nightinagle(x)

mammal group

- If mammal(x) $\wedge$ long neck(x) $\wedge$ four legs(x) then giraffe(x)
- If mammal(x) $\wedge$ swims(x) then dolphin(x)

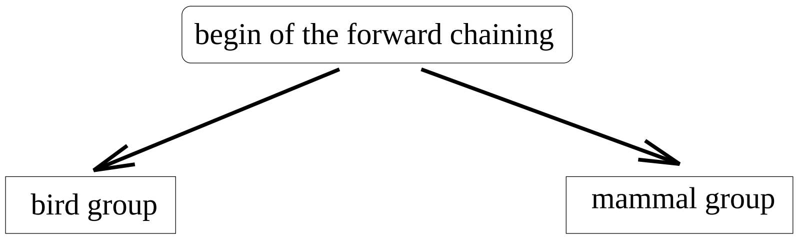


Figure 4.1: Semantic net representing the taxonomy of the rules about birds and mammals.

Uncertainty can be expressed by the certainty theory. The heuristic function can be represented by the CF values of the conclusion patterns in the case when conflict resolution is used. Deduction systems are often used by expert systems which were developed with the engineering aim to capture the knowledge of an expert in a certain closed domain [110, 167, 76]. The type of knowledge captured by these expert systems was often organized by human society in different kinds of taxonomies, for example taxonomy of animals, or the taxonomy of medical illness.

The semantic net representation of the taxonomy of rules is related to the decision tree approach [160]. A decision tree is used to determine a category by a function that maps each element of its domain to a category label or numerical value. At each leaf of a decision tree, one finds a category label. It is determined at nodes by tests that have a small number of possible outcomes. A decision tree with a range of symbolic category or class labels is called a classification tree. In our approach, the test are described by rules. A decision tree with a range of continuous numeric values is called a regression tree. Decision trees show clearly how to reach a decision. They are constructed automatically by C4.5 [161] and CART (Classification and Regression Tree) [26] programs. Decision trees can be also converted into rules by the C4.5 program.

There is also a connection between neural networks and decision trees. Feed forward neural networks can be initialized by decision trees [75]. In this case decision trees represent some approximation of the target concept which helps to determine the initial weight setting and the architecture of the feedforward neural network. The architecture is determined by the number of neurons and the topology of connections between them. This kind of initialization was proven successful in different real world applications [90, 91].

4.3 Hierarchical Categorization

4.3.1 Neural deductions system

Hierarchical categorization is performed by a neural deductive system. The model is composed of connected associative memories which represent groups of rules. The problem space is represented by connections between the associative memories and those connections correspond to conclusions of the rules. The known features are represented in an external memory before the forward chaining begins (see fig. 4.2). This model is appropriate for taxionomical representation of knowledge with the aid of verbal categories. Categories can be divided into subcategories, so that a taxonomy can be constructed and represented by an acyclic graph. The nodes in this graph correspond to categories and the links indicate the "is a subcategory" relation between them. The process of the hierarchical categorization can be performed by moving from more general categories to more specific categories until the desired categories are reached. The rules in each associative memory are represented as described in chapter 3 with the exception that the categories which are represented by the conclusion may be divideed into more exact subcategories by another associative memory (see fig. 4.2). The associative memory with the name of the corresponding category which it divided is called a module. A disjunction of patterns in a premise can be represented by two different rules. Negation can not be interpreted, only represented by a corresponding feature name.

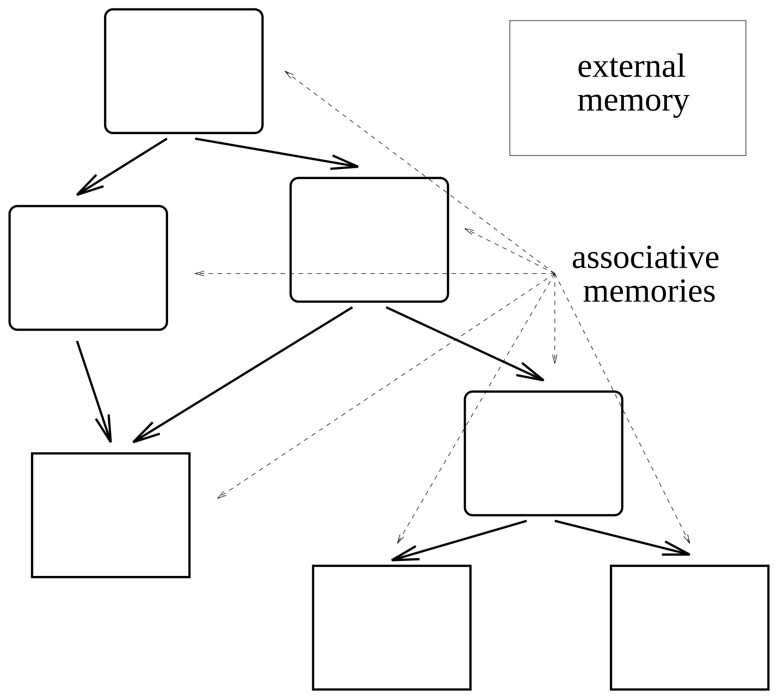


Figure 4.2: Neural deductions system.

4.3.2 Representation by associative memories

A knowledge base for diagnosing car problems is represented by a taxonomy [223]. The forward chaining for the determination of a disorder begins at the root category *problem* which is divided into other subcategories. The first pattern of the premise represents the name of the category which is divided:

If **problem** $\wedge$

1. noise or problem when car is standing(3) then other
2. problem during start(2) then starter
3. electrical problem(2) then with [0.33] engine is dead
4. electrical problem(2) then electrical
5. problem during driving(3) then driving
6. noise or problem when car is standing(3) then with [0.5] engine
7. noise or problem when car is standing(3) then with [0.5] radiator

Subcategories are divided again in subcategories, for example the subcategory driving:

If **driving** $\wedge$

1. mechanical problem(4) then with [0.6] other
2. difficulties with engine(4) then with [0.33] engine starts
3. oil pressure lamp lights up(3) then oil
4. difficulties with engine(4) then engine
5. difficulties with radiator(4) then radiator

And in subcategories which describe the disorder:

If **engine** $\wedge$

1. smoke in the engine compartment(5) then radiator bad or coolant tube bad
2. smoke in the engine compartment(5) then with [0.67] short curcuit
3. poor engine performance(6) then dirty air filter or bad spark plugs
4. poor engine performance(6) then with [0.5] fuel pump bad or fuel pipe broken
5. idling irregular or engine comes to standstill(5) then bad spark plugs or failure in the ignition system
6. idling irregular or engine comes to standstill(5) then with [0.67] compression is insufficient
7. a lot of steam from the exhaust(5) then cylinder casket broken

Another group of subcategories which describe disorders:

If **radiator** $\wedge$

1. bubble in coolant compensation tank(4) then water pump broken
2. engine gets hot(5) then bad water pump or bad thermostat
3. engine gets hot(5) then with[0.6] bad fan or loss of coolant

The knowledge base is composed of 48 rules which are represented by 12 moduls (see fig. 4.3).

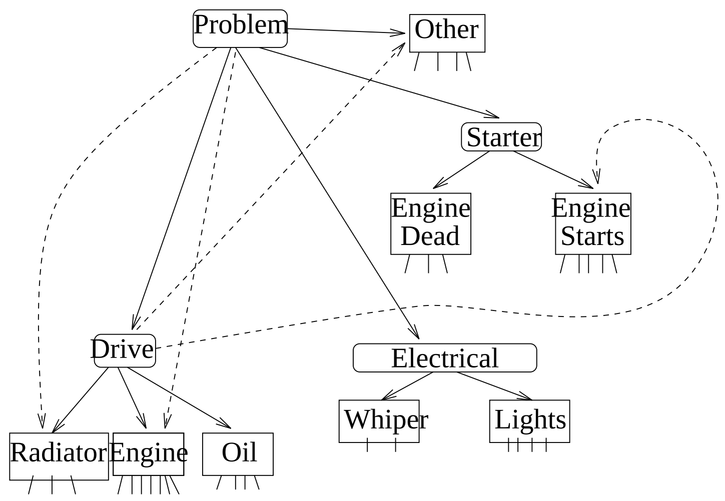


Figure 4.3: Taxonomic representation of disorders of a car. Uncertain categories are represented by dotted arrows.

4.3.3 Categorization

The external memory guides the search in the problem space in the direction of the most plausible category, which is the category with the highest quality criterion value. In the search the first favored category can turn out to be wrong because it is assumed that its subcategories are not present. In this case another category is examined. This search strategy corresponds to hill climbing [229], which is a depth-first search in which the choices are ordered according to the quality criteria values (see fig. 4.4). The quality criteria values represent a heuristic measurement of the distance to the remaining goal.

Example

The observer specifies the following features using the belief table 1.4:

1. probably **problem during driving**
2. probably not **electrical problem**
3. very probably **engine gets hot**

The forward chaining begins at the module problem. The quality criteria of the categories in each module are determined by the associative memory as in paragraph 1.2. The overall quality criteria of the category which is represented by l chained rules is given by the mean of the values computed by the associative memories $m \in \{1, 2.., l\}$:

$$qc(Ca_i) = \frac{\sum_{m=1}^{l} qc(Ca_{i(m)})}{l} =$$

$$= \frac{\sum_{m=1}^{l} \left(\frac{2}{\sum_{j(m)=1}^{n(m)} \delta(w_{i(m)j(m)})} \cdot \sum_{j=1}^{n(m)} \delta(w_{i(m)j(m)} x_{j(m)}) \right)}{l} - 1$$

The categorization begins at the module *problem*:

$$qc(other) = -0.33,\ qc(starter) = 0,\ qc(engine\ is\ dead) = -1$$

$$qc(electrical) = -1,\ qc(driving) = 0.33,\ qc(engine) = -0.5,\ qc(radiator) = -0.5$$

The subcategory with the highest qc value is chosen. The chosen subcategory *driving* is divided:

$$qc(other_{module}) = -0.2,\ qc(engine\ starts_{module}) = -0.33,\ qc(oil_{module}) = -0.33$$

$$qc(engine_{module}) = 0,\ qc(radiator_{module}) = 0$$

The category *driving* and *engine* have the same qc_{module} values. Module engine is randomly chosen. The qc value is

$$qc(engine) = \frac{qc(driving) + qc(engine_{module})}{2} = 0.17$$

The chosen subcategory *engine* is divided:

$$qc(radiator\ bad\ or\ coolant\ tube\ bad_{module}) = -0.2,\ qc(short\ curcuit_{module}) = -0.33$$

$$qc(dirty\ air\ filter\ or\ bad\ spark\ plugs_{module}) = 0$$

$$qc(fuel\ piump\ bad\ or\ bad\ spark\ plugs_{module}) = -0.25$$

$$qc(bad\ spark\ plugs\ or\ falure\ in\ the\ ignition\ system_{module}) = -0.2$$

$$qc(compression\ is\ insufficient_{module}) = -0.33$$

$$qc(cylinder\ casket\ broken_{module}) = -0.2$$

The low qc_{module} values indicate a possible absence. The search is continued. Backtracking is carried out and the qc_{module} value of the subcategory *engine* is set to -1[1]:

$$qc(other_{module}) = -0.2,\ qc(engine\ starts_{module}) = -0.33,\ qc(oil_{module}) = -0.33$$

[1]That way examined categories are not re-examined by another path.

$$qc(engine_{module}) = -1,\ qc(radiator_{module}) = 0$$

The subcategory with the highest qc_{module} value is chosen. The chosen subcategory is now *radiator*.

$$qc(raiator) = \frac{qc(driving) + qc(radiator_{module})}{2} = 0.17$$

The chosen subcategory *radiator* is divided:

$$qc(the\ water\ pump\ broken_{module}) = 0$$

$$qc(bad\ water\ pump\ or\ bad\ thermostat_{module}) = 0.6$$

$$qc(bad\ fan\ or\ losss\ of\ coolant_{module}) = 0.33$$

The qc_{module} values of two categories indicate possible presence:

$$qc(bad\ water\ pump\ or\ bad\ thermostat) = 0.31 =$$

$$= \frac{qc(driving) + qc(radiator_{module}) + qc(bad\ water\ pump\ or\ bad\ thermostat_{module})}{3}$$

$$= \frac{qc(radiator) \cdot (3-1)}{3} + \frac{qc(bad\ water\ pump\ or\ bad\ thermostat_{module})}{3}$$

and

$$qc(bad\ fan\ or\ losss\ of\ coolant) = 0.22 =$$

$$= \frac{qc(driving) + qc(radiator_{module}) + qc(bad\ fan\ or\ losss\ of\ coolant_{module})}{3}$$

$$= \frac{qc(radiator) \cdot (3-1)}{3} + \frac{qc(bad\ fan\ or\ losss\ of\ coolant_{module})}{3}$$

The barrier in our system which determines when a category is treated as possibly absent dependent on the qc_{module} value is represented by two constants. If the qc_{module} value is below ω for a category which is divided, or below ϖ for a category which is not divided, backtracking occurs. The minimal qc value for a category which is not divided is above qc_{min}. It is dependet on the deepness of the hierarchical categorization:

$$qc_{min} = \frac{\omega \cdot (deep - 1)}{deep} + \frac{\varpi}{deep.}$$

Depending on the value of the constants, more certain or less certain categories are output by the system. The extreme example is the output of the whole knowledge base for $\omega = -1, \varpi = -1$. In our examples we set $\omega = -0.5\ and\ \varpi = 0.2$.

A star * represents the presence of features, the value between "<" and ">" gages the hierarchy of the categorization:

```
**
1) ! MODULE DRIVING with qc=0.33, <1> !
2) ! MODULE ENGINE with qc=0.17, <2> !
3) ! MODULE RADIATOR with qc=0.17, <2> !*
---------------------------------------------
R E S U L T :
```

BAD WATER PUMP OR BAD THERMOSTAT with qc=0.31
BAD FAN OR LOSS OF COOLANT with qc=0.22

4.3.4 Availability heuristics

Only one feature is known to the observer:

1. certainly **poor engine performance**

```
1) ! MODULE STARTER with qc=0, <1> !
2) ! MODULE ENGINE_STARTS with qc=0, <2> !
3) ! MODULE ENGINE_IS_DEAD with qc=-0.17, <2> !

4) ! MODULE ELECTRICAL with qc=0, <1> !
5) ! MODULE LIGHTS with qc=0, <2> !
6) ! MODULE WIPER with qc=0, <2> !

7) ! MODULE OTHER with qc=-0.33, <1> !
8) ! MODULE DRIVING with qc=-0.33, <1> !
9) ! MODULE ENGINE with qc=-0.17, <2> !*
---------------------------------------------
R E S U L T :
```

DIRTY AIR FILTER OR BAD SPARK PLUGS with qc=0.22
FUEL PUMP BAD OR FUEL PIPE BROKEN with qc=0.06

The information was sufficient for the determination of the disorder but insufficient to guide the search, which took nine steps. During the hierarchical categorization by humans some categories come to mind more easily, because they were determined to be more frequent than the other. Some psychologists [213] assume that the individual estimates the frequency of an event, in our case the determination of a category. This kind of heuristic is called the availability heuristic [213, 226, 168]. The idea of how to implement this kind of heuristic was already given by the psychologist William James around 1870 [78], page 4: *"Habits are due to pathways by the nerve centers, in getting out they leave their traces in* *the paths which they take"*. In our model, this corresponds to the strengthening of the links between the categories which describe a successful search by a small factor. The successful search corresponds to the path of links between the categories from the category where the search began to the results of the hierarchical categorization. Paths that corresponds to wrong search directions are weakened by a small factor ϵ, see fig. 4.4. All links are initialize to zero, and the learning rule is:

$$\textit{link to category}^{new} = \textit{link to category}^{old} + \epsilon \qquad \textit{if present.}$$

$$\textit{link to category}^{new} = \textit{link to category}^{old} - \epsilon \qquad \textit{if backtracking.}$$

$$\textit{link to category}^{new} = \textit{link to category}^{old} \qquad \textit{else.}$$

During the repeated search the categories that receive strong links are favored. The search direction corresponds to the highest value that is the sum of the quality criterion of a category and the strength of the link to it. In terms of bayesian statistics the chosen category has the highest posterior probability which is composed of actual likelihood and prior probability [152, 110].

Suppose that we are trying to diagnose the problems with cars that are used near a dusty desert where sandstorms are frequent. Very often the problems of such cars are caused by dirty air filters. After strengthening the corresponding links between the categories during the previous categorizations which describe this fact [2], the observer specifies one feature:

1. very probably **poor engine performance**

```
1) ! MODULE DRIVING with qc=-0.33, <1> !
2) ! MODULE ENGINE with qc=-0.17, <2> !*
---------------------------------------------
R E S U L T :
```

DIRTY AIR FILTER OR BAD SPARK PLUGS with qc=0.11

The search takes only two steps with the availability heuristic after learning.

[2] $frequency = 5, \epsilon = 0.1$

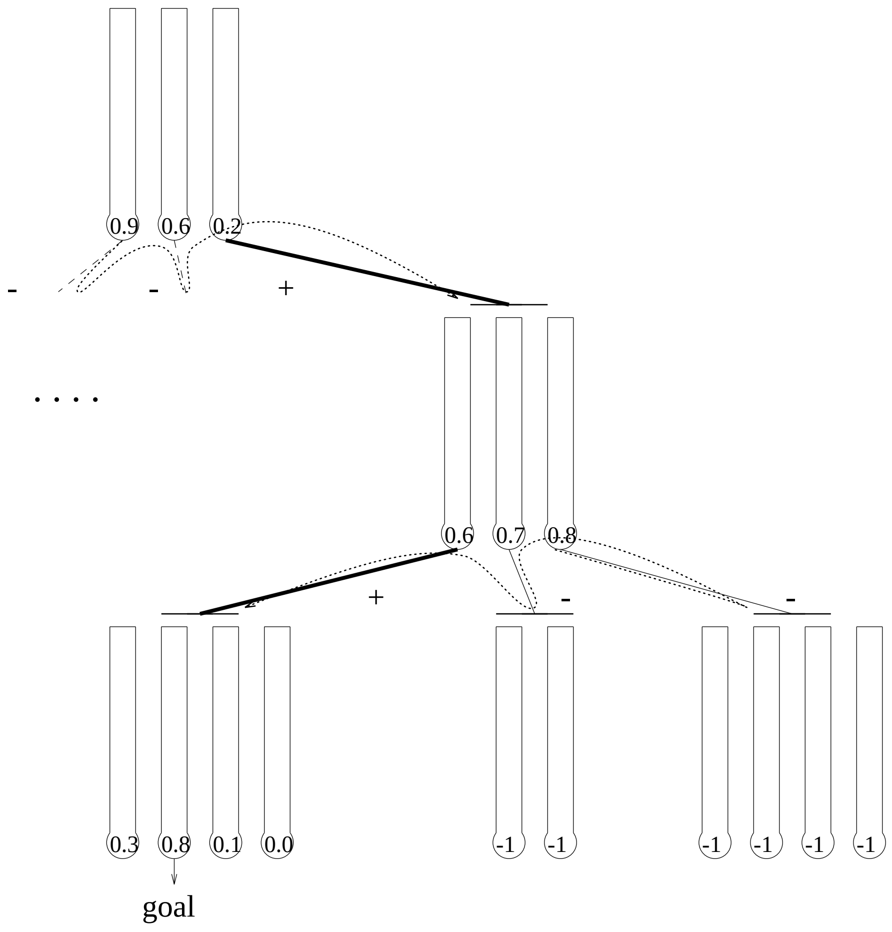


Figure 4.4: The learning of the availability heuristic. Links between the categories which describe a successful search are strengthened by a small factor, shown by +. Links to categories where backtracking occurs are weakened by a small factor, shown by -.

4.3.5 Implementation

Each module is composed of an associative memory in which the correlation between the features and corresponding categories is stored, and an *awi* for input operation (see fig. 2.5 and paragraph 2.2.1 about word recognition). In the external memory the corresponding feature names are represented by strings, together with belief values named by adjectives. During the categorization each feature name in the external memory is tested by the *awi* of the module. If it is recognized despite possible type errors, the belief value is calculated using the corresponding belief table (see paragraph 1.2.4). Otherwise the belief in the not specified is calculated (see also paragraph 1.2.4).

4.3.6 Query-Reply

If the categorization must be as accurate as possible, the presence of each category in a module is tested by queries about presence of corresponding features [41, 83, 163]. This procedure corresponds to the abductive inference in which a hypothesis is made and then is tested [153, 81]. The hierarchical categorization is performed as before except that the search is now guided by an observer instead of the external memory. An example of a dialog of a medical instruction system for how to revive a person is shown [3]. It follows the guidelines of the American Heart Association [109], the taxonomy is shown in Fig. 4.5. During the categorization the instructions about the present medical treatment are shown [224].

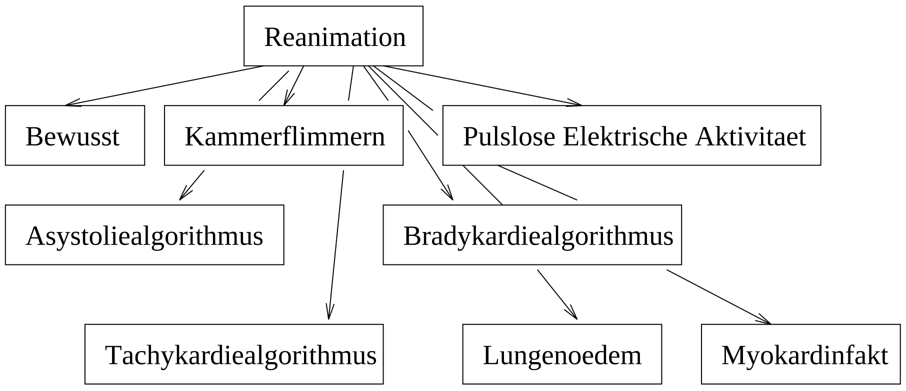


Figure 4.5: Medical treatment of reanimation.

In following example a dialogue between a student and the system is shown:

! MODULE REANIMATION !
CONSULTING: input NO, MAYBE, YES.

Question Nr.1, BEWUSST present? >> NO
Question Nr.2, BEWUSSTLOS present? >> YES
• Alarmiere den Rettungsdienst
• Fordere eine Defibrillator an
• Ueberpruefe die Atmung (Oeffnen der Atemwege, Sehen, Hoeren und Fuehlen)
Question Nr.3, SPONTANATMUNG present? >> NO

[3]This system is in German to make the usage easier for German students.

Question Nr.4, ATEMSTILLSTAND present? >> YES
• Beatme 2mal langsam
• Ueberpruefe den Kreislauf
Question Nr.5, KEIN PULS present? >> YES
• Beginne mit CPR
Question Nr.6, KAMMERFLIMMERN ODER VENTIKULAERE TACHYKARDIE IM EKG present? (0 - 2) >> MAYBE
• Ueberpruefe ABC
• Beginne mit CPR bis Defibrillator verfuegbar
• KF/VT im EKG
• Defibrilliere - wenn noetig - bis 3 mal bei anhaltendem
• KF/VT (200J, 200-300 J, 360J)
Question Nr.7, KEIN KAMMERFLIMMERN ODER VENTIKULAERE TACHYKARDIE IM EKG present? >> NO
Question Nr.8, ELEKTRISCHE AKTIVITAET present? >> MAYBE
Question Nr.9, KEINE ELEKTRISCHE AKTIVITAET present? >> NO
Question Nr.10, PULS present?, >> NO
Question Nr.11, ARRHYTHMIE ZU LANGSAM present? >> MAYBE
• Ueberpruefe ABC
• Sichere Atemwege
• Gib Sauerstoff
• Lege einen i.v.-Zugange
• Sorge fuer EKG-Monitoring, Pulsoximetrie, automatische Blutdruckmessung
• Ueberpruefe die Vitalzeichen
• Erhebe eine Kurzanamnese
• Untersuche den Patienten
• Veranlasse ein 12-Kanal-EKG
• Veranlasse eine Roentgen-Thorax-Aufnahme, wenn moeglich am Ort
• BRADYKARDIE: absolut ($<$ 60 Schlaege/min) oder relativ
Question Nr.12, ARRHYTHMIE ZU SCHNELL present? >> NO

! MODULE KAMMERFLIMMERN with qc=0.4 !
Question Nr.13, RHYTMUS NACH DEN ERSTEN DREI SCHOCKS ANHALTENDES ODER WIEDERAUFTRETENDES KF VT present? >> NO
Question Nr.14, RHYTMUS NACH DEN ERSTEN DREI SCHOCKS SPONTANKREISLAUF present? >> MAYBE

R E S U L T :
UEBERPRUEFE DIE VITALZEICHEN UNERSTUETZE ATMUNG
• Ueberpruefe die Vitalzeichen
• Oeffne die Atemwege
• Unterstuetze die Atmung
• Gib ggf. Medikamente zur Stabilisierung von Blutdruck, Herzfrequenz und -rhythmus.

with qc=0.2

By answering the questions with three possible answers expressing the certainty of the presence of the present symptoms and corresponding instructions, the student learns, in a playful way, the taxonomic knowledge, knowledge which is dependet on the presence of symptoms. The main advantage of this system compared to other systems is its clarity and the easy of use. To learn knowledge which is represented by a system, the behavior of the system must be understood. The question can be answered by only three adjectives. The student can understand the treatment of uncertainty by counting and is not obliged to learn a (complicated) mathematical uncertainty calculus paradigm.

4.4 System Jurassic

The goal of the system is to help paleontologists determine creatures or creature groups out of a taxonomic knowledge base which describes the dinosaurs [62] and using only some vague beliefs about the presence and absence of some features. In 1887 Professor Harry Govier Seeley grouped all dinosaurs into the saurischia and ornithischia groups according to their hip design. The saurischian were divided later into two subgroups: the carnivorous, bipedal theropods and the plant-eating, mostly quadruped sauropodomorphs. The ornithischians were divided into the subgroups birdlike ornithopods, armored thyreophorans, and margginoncephalia. The subgroups can be divided into suborders and then into families and finally into genus. The genus includes the species. It must be noted that in this taxonomy many relations are only guesswork, and many paleontologists have different ideas about how the taxonomy should look. The whole knowledge base is composed of 70 modules in which 423 rules are stored. The taxonomy is modeled in the books of [101, 102] in which over fifty families and more than 340 genre of dinosaurs are described.

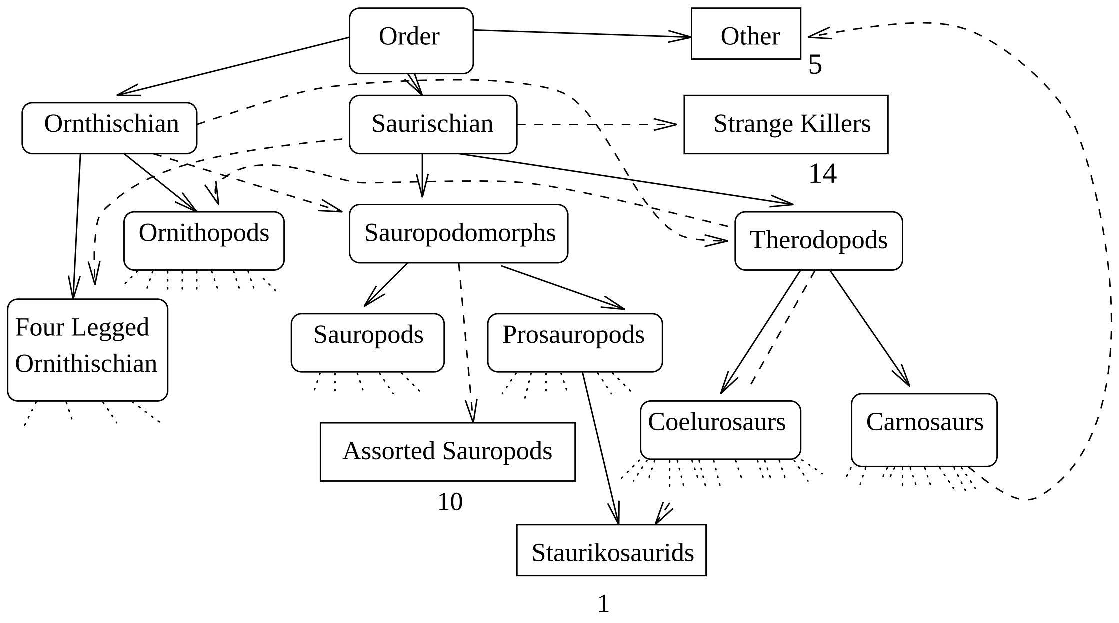


Figure 4.6: Taxonomy of dinosauria. The number written below the rectangular boxes represents the categories which are not divided, the species. Uncertain categories are represented by dotted arrows.

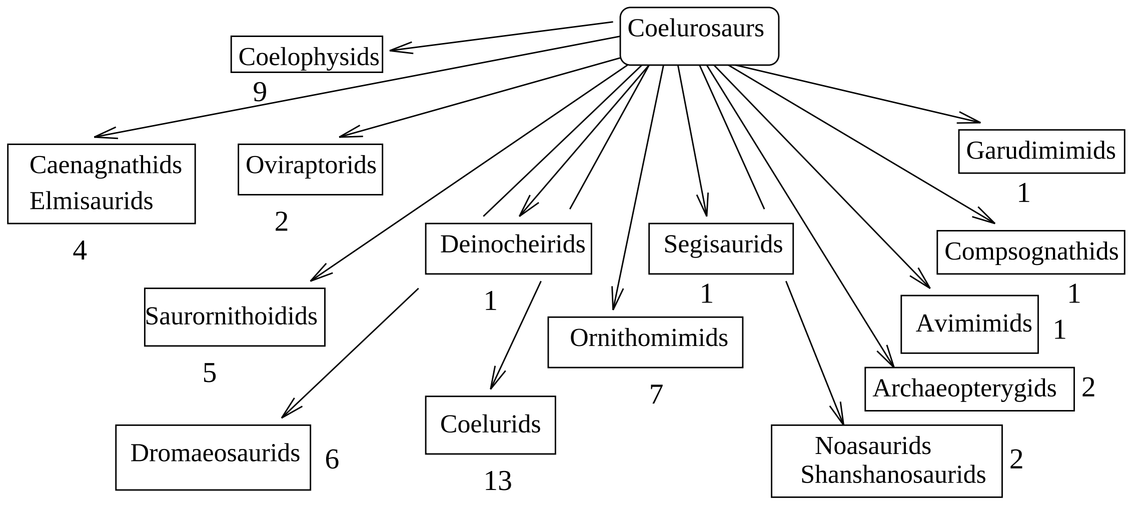


Figure 4.7: Infraorder Coelurosaurs.

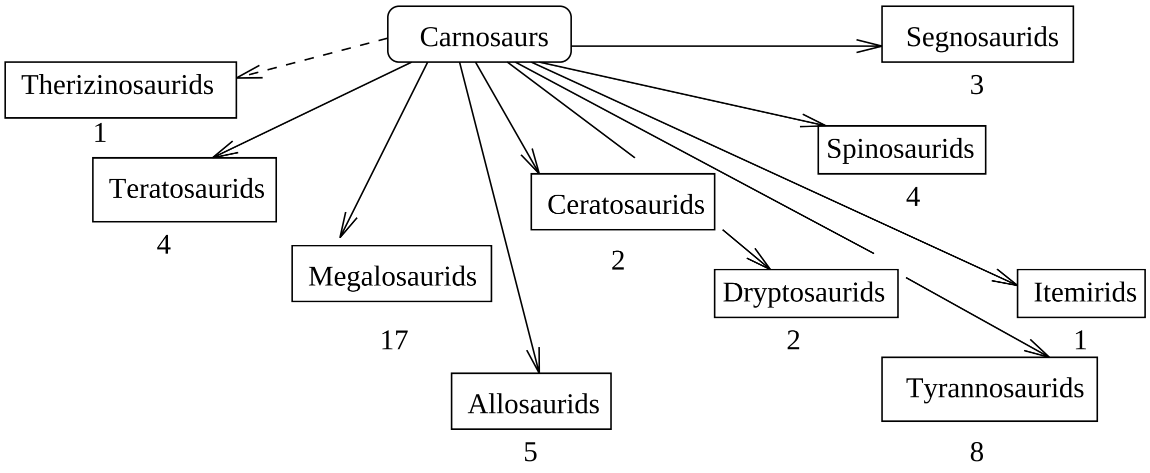


Figure 4.8: Infraorder Carnosaurus.

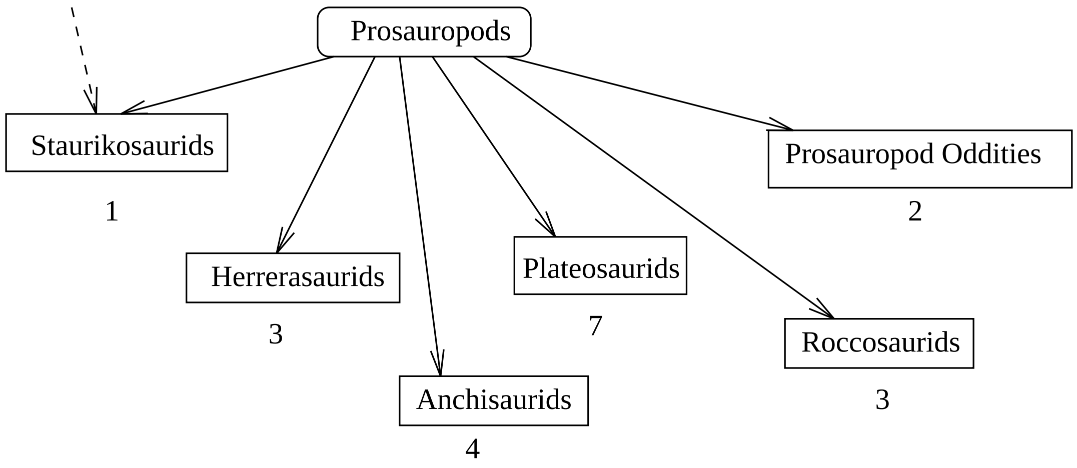


Figure 4.9: Infraorder Prosauropods.

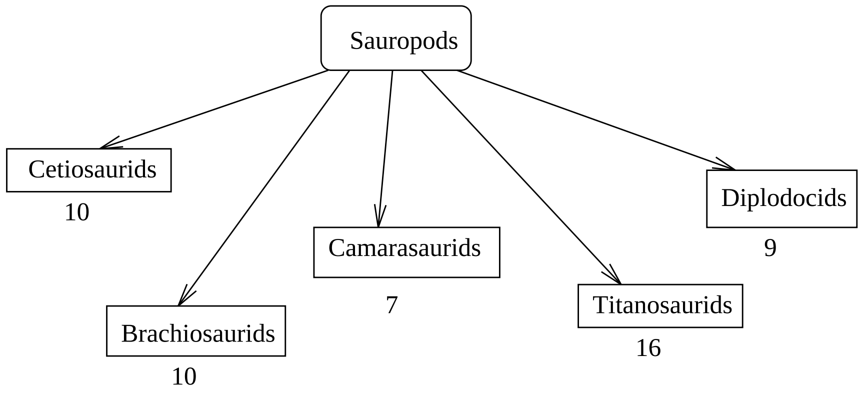


Figure 4.10: Infraorder Sauropods.

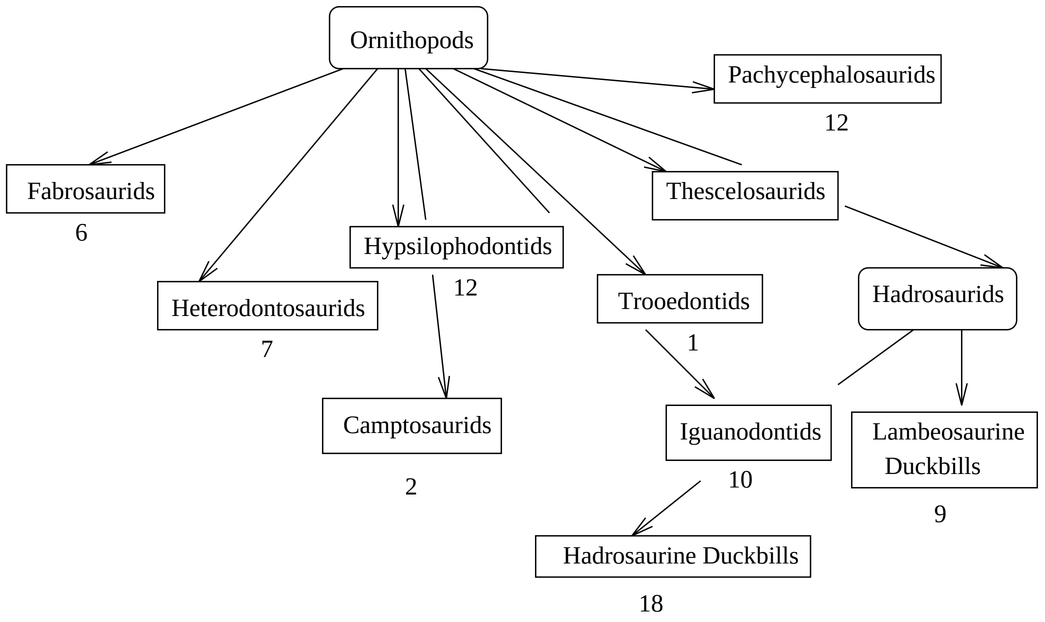


Figure 4.11: Infraorder Ornithopods.

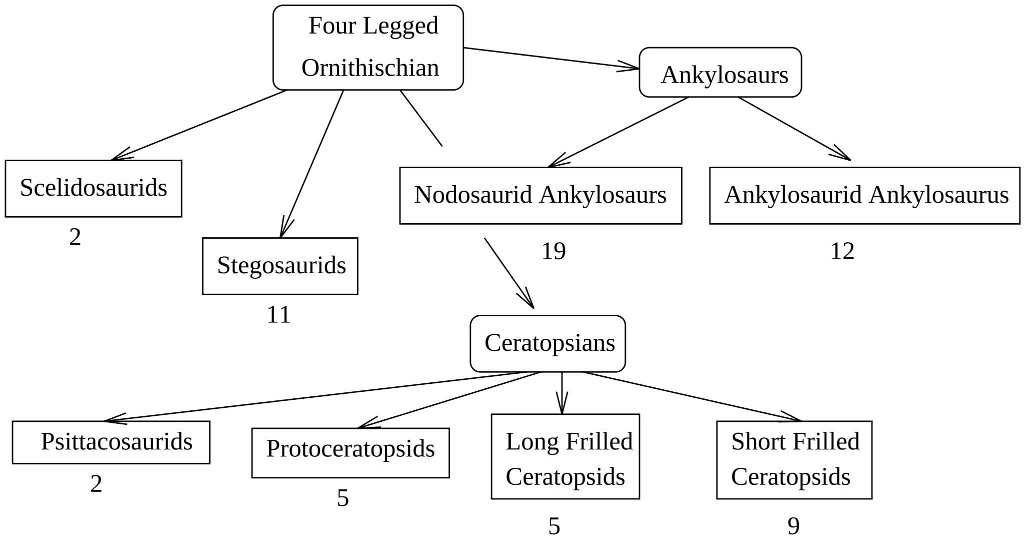


Figure 4.12: Infraorder Four Legged Ornithishian.

4.4.1 Categorization

In the following example the retrieval of taxonomic knowledge from a database, which tries to completely describe a knowledge area, is shown. The species of fig. 4.13 is described by six features using the belief table.

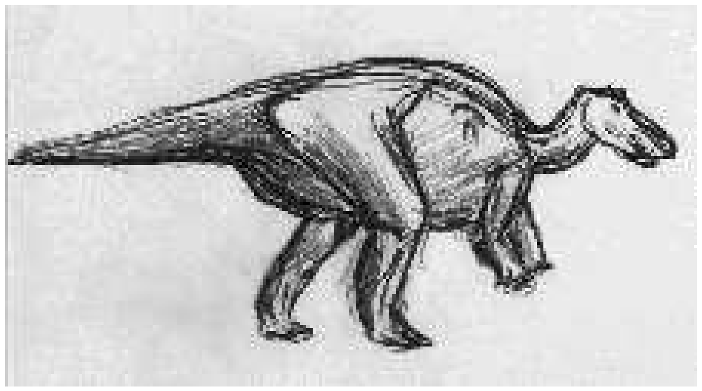


Figure 4.13: Maiasaura.

1. probably **bird hipped**
2. certainly **two legged**
3. probably **long arms**

4. certainly **long stiffly held tail**

5. probably **solid bony humps or crests**

6. very probably **length nine m**

```
*
1) ! MODULE ORNITHISCHIAN with qc=0.33, <1> !*
2) ! MODULE ORNITHOPODS with qc=0.67, <2> !**
3) ! MODULE PACHYCEPHALOSAURIDS with qc=0.51, <3> !
4) ! MODULE THESCELOSAURIDS with qc=0.49, <3> !
5) ! MODULE HADROSAURIDS with qc=0.48, <3> !*
6) ! MODULE HADROSAURINE_DUCKBILLS with qc=0.36, <4> !*
----------------------------------------------
R E S U L T :
```

MAIASAURA with qc=0.38
SAUROLOPHUS with qc=0.33

The first determined categoriy represents *Maiasaura* (see fig. 4.13). *Maiasaura* and *Saurolophus* belongs to the group *Hadrosaurine duckbills*[4].

In chapter 1 another example of hierarchical categorization was shown.

4.4.2 Hypothesis

Which object of one group is most similar to an object of another, different group? For example, we have two different disjunct knowledge areas which are completely described. The two groups are dinosaurs and mammals. We wish to know which animal of the dinosaur group is most similar to an animal of the mammal group. This is a kind of case-based reasoning [64, 170] in which specific knowledge is used to retrieve the most similar stored case (see also section 5.1).

Suppose we pose a question: Which dinosaur species is most similar to a human? We describe a human being by seven features. The hierarchical categorization is performed in which the most similar stored category is determined.

1. certainly **two legged**

2. certainly not **four legged**

3. probably **thin walled fragile bones**

4. very probably not **big**

5. very probably **big eyes**

[4]For continued search see Appendix A.3.1

6. certainly **big brain**
7. certainly **capsule in the skull**
8. very probably **length two m**

```
1) ! MODULE SAURISCHIAN with qc=-0.33, <1> !**
2) ! MODULE THEROPODS with qc=0.05, <2> !*
3) ! MODULE COELUROSAURS with qc=0.12, <3> !***
4) ! MODULE SAURORNITHOIDIDS with qc=0.2, <4> !*
----------------------------------------------
R E S U L T :
```

SAURORNITHOIDES with qc=0.25
STENONYCHOSAURUS with qc=0.25

This answer is the same as the suggestion of Dale Russell [165] in the early 1980s that the Stenonychosaurus, see fig 4.14 (also now known as Troodon), could have given rise to a brainy descendant, had dinosaurs survived instead of dying out [101, 102][5].

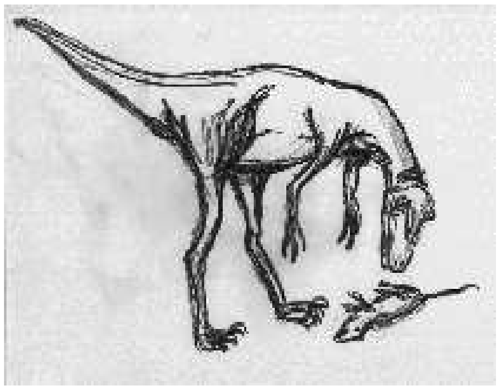


Figure 4.14: Stenonychosaurus.

4.4.3 Priming

A paleontologist tries to determine a species from vague knowledge by forming possible hypotheses. He specifies the most probable species by declarative knowledge in the form of some of the most noticeable features and the species is determined. At the same time the corresponding taxonomic knowledge area

[5]For continued search see Appendix A.3.2

is primed by non-declarative knowledge with the aid of the availability heuristic. After this the paleontologist formulates more sophisticated features and the species of the primed taxonomic area are examined first. Thus, priming eases the formation of the final hypothesis, as more exact, possible hypotheses are formed.

Priming of knowledge areas [131, 8, 192] is performed by one pass strengthening and weakening of the links between categories.[6] This is accomplished by utilizing a sufficient factor[7] to permit the retrieval of the same part of the taxonomy with insufficient knowledge by the usage of the availability heuristic during the next categorization. In this example we determine a species which had a very thick skull:

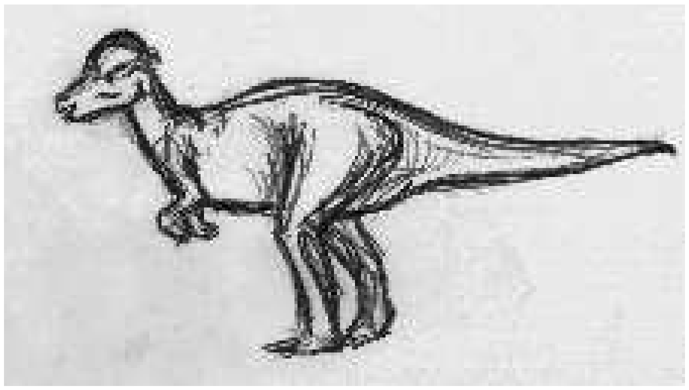


Figure 4.15: Pachycephalosaurus.

1. certainly **skull bone has a twenty five cm thick roof**

```
1) ! MODULE SAURISCHIAN with qc=-0.33, <1> !
2) ! MODULE THEROPODS with qc=-0.24, <2> !
3) ! MODULE COELUROSAURS with qc=-0.16, <3> !
4) ! MODULE CAENAGNATHIDS_ELMISAURIDS with qc=-0.12, <4> !
5) ! MODULE OVIRAPTORIDS with qc=-0.12, <4> !
6) ! MODULE SEGISAURIDS with qc=-0.12, <4> !
7) ! MODULE COMPSOGNATHIDS with qc=-0.12, <4> !
8) ! MODULE DROMAEOSAURIDS with qc=-0.14, <4> !
9) ! MODULE GARUDIMIMIDS with qc=-0.14, <4> !
10) ! MODULE ORNITHOMIMIDS with qc=-0.14, <4> !
11) ! MODULE SAURORNITHOIDIDS with qc=-0.14, <4> !
12) ! MODULE ARCHAEOPTERYGIDS with qc=-0.15, <4> !
13) ! MODULE COELOPHYSIDS with qc=-0.15, <4> !
```

[6]see paragrap 4.3.4

[7]$\epsilon = 0.5$

```
14) ! MODULE COELURIDS with qc=-0.16, <4> !
15) ! MODULE NOASAURIDS_SHANSHANOSAURIDS with qc=-0.17, <4> !
16) ! MODULE DEINOCHEIRIDS with qc=-0.17, <4> !
17) ! MODULE AVIMIMIDS with qc=-0.2, <4> !

18) ! MODULE CARNOSAURS with qc=-0.2, <3> !
19) ! MODULE SEGNOSAURIDS with qc=-0.16, <4> !
20) ! MODULE SPINOSAURIDS with qc=-0.17, <4> !
21) ! MODULE CERATOSAURIDS with qc=-0.18, <4> !
22) ! MODULE DRYPTOSAURIDS with qc=-0.18, <4> !
23) ! MODULE TYRANNOSAURIDS with qc=-0.18, <4> !
24) ! MODULE MEGALOSAURIDS with qc=-0.18, <4> !
25) ! MODULE ALLOSAURIDS with qc=-0.18, <4> !
26) ! MODULE TERATOSAURIDS with qc=-0.2, <4> !
27) ! MODULE OTHER with qc=-0.23, <4> !
28) ! MODULE THERIZINOSAURIDS with qc=-0.23, <4> !

29) ! MODULE ORNITHOPODS with qc=-0.23, <3> !
30) ! MODULE THESCELOSAURIDS with qc=-0.18, <4> !
31) ! MODULE PACHYCEPHALOSAURIDS with qc=-0.19, <4> !*
----------------------------------------------
R E S U L T :
```

PACHYCEPHALOSAURUS with qc=-0.09

Pachycephalosaurus, "Thick-headed lizard" (see fig. 4.15) is one species from the group of *Pachycephalosaurids* which are supposed to be the equivalent of today's bighorn sheep, (see fig. 4.16) *"In the mating season, big mammals evidently ran after one another, clashing heads together to decide which would dominate and mate with the whole herd of females"* [102], page 154.

We search another species with "thick rough shaped dome":

1. certainly **skull with a thick rough dome shaped top**

```
1) ! MODULE SAURISCHIAN with qc=-0.33, <1> !
2) ! MODULE THEROPODS with qc=-0.24, <2> !
3) ! MODULE ORNITHOPODS with qc=-0.23, <3> !
4) ! MODULE PACHYCEPHALOSAURIDS with qc=-0.19, <4> !*
----------------------------------------------
R E S U L T :
```

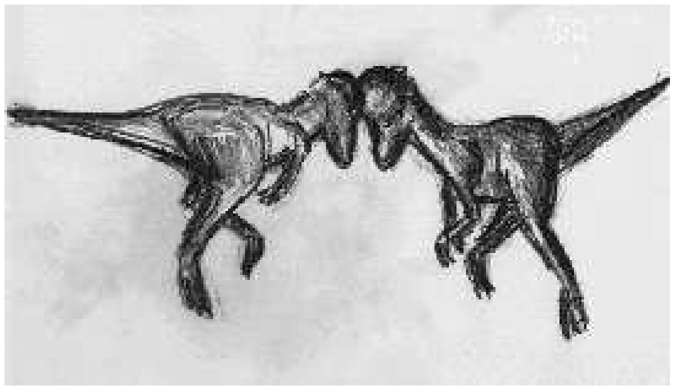


Figure 4.16: “Thick-headed lizard” ran one another, clashing heads together.

MAJUNGATHOLUS with qc=-0.11

The search leads directly to the right group *Pachycephalosaurids*. The links between the categories are initialized to zero and another group is primed. The roof lizard of fig. 4.17 is described by three features, it is a stegosaurid with the largest known plates:

1. probably **bird hipped**
2. certainly **four heavy spikes**
3. certainly **plates up to seventy six cm high and seventy nine cm long**

```
*
1) ! MODULE ORNITHISCHIAN with qc=0.33, <1> !
2) ! MODULE ORNITHOPODS with qc=0, <2> !
3) ! MODULE THESCELOSAURIDS with qc=-0.02, <3> !
4) ! MODULE PACHYCEPHALOSAURIDS with qc=-0.02, <3> !
5) ! MODULE HYPSILOPHODONTIDS with qc=-0.04, <3> !
6) ! MODULE FABROSAURIDS with qc=-0.04, <3> !
7) ! MODULE HADROSAURIDS with qc=-0.04, <3> !
8) ! MODULE HADROSAURINE_DUCKBILLS with qc=-0.08, <4> !
9) ! MODULE LAMBEOSAURINE_DUCKBILLS with qc=-0.09, <4> !

10) ! MODULE CAMPTOSAURIDS with qc=-0.04, <3> !
11) ! MODULE HETERODONTOSAURIDS with qc=-0.05, <3> !
12) ! MODULE TROOEDONTIDS with qc=-0.05, <3> !
13) ! MODULE IGUANODONTIDS with qc=-0.05, <3> !

14) ! MODULE FOUR_LEGGED_ORNITHISCHIANS with qc=0, <2> !
```

```
15) ! MODULE CERATOPSIANS with qc=-0.02, <3> !
16) ! MODULE PSITTACOSAURIDS with qc=-0.05, <4> !
17) ! MODULE PROTOCERATOPSIDS with qc=-0.05, <4> !
18) ! MODULE SHORT_FRILLED_CERATOPSIDS with qc=-0.06, <4> !
19) ! MODULE LONG_FRILLED_CERATOPSIDS with qc=-0.07, <4> !

20) ! MODULE STEGOSAURIDS with qc=-0.05, <3> !**
----------------------------------------------
R E S U L T :
```

STEGOSAURUS with qc=0.1

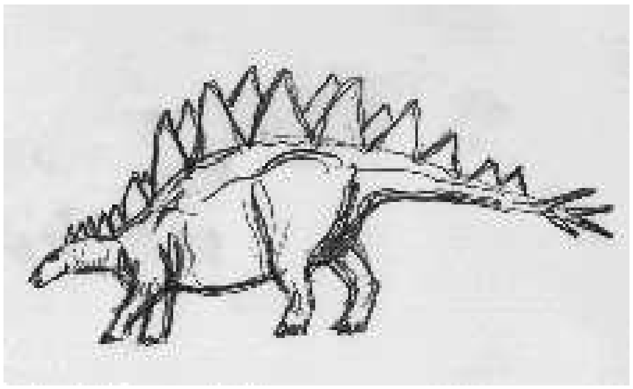


Figure 4.17: Stegosaurus.

Species with triangular plates which lived in the late Jurassic are searched:

1. very probably not **lizard hipped**
2. certainly **late jurassic**
3. certainly **triangular plates**

```
*
1) ! MODULE ORNITHISCHIAN with qc=-0.33, <1> !
2) ! MODULE FOUR_LEGGED_ORNITHISCHIANS with qc=-0.33, <2> !
3) ! MODULE STEGOSAURIDS with qc=-0.27, <3> !**
----------------------------------------------
R E S U L T :
```

KENTROSAURUS with qc=-0.04
TUOJIANGOSAURUS with qc=-0.04
DACENTRURUS with qc=-0.12

Dacentrurus has the lowest *qc* value, it lived in the middle to late Jurassic and had perhaps no plates.

Recovering

The links between the categories are not initialized and a species from another group is searched. It is *Parasaurolophus* of the "Lambe's lizards" group (see fig 4.18). The tube inside a *"Parasaurolophus*'s *crest acted as a sound box, amplifying the voice and producing low, resonant cries"* [102], page 150.

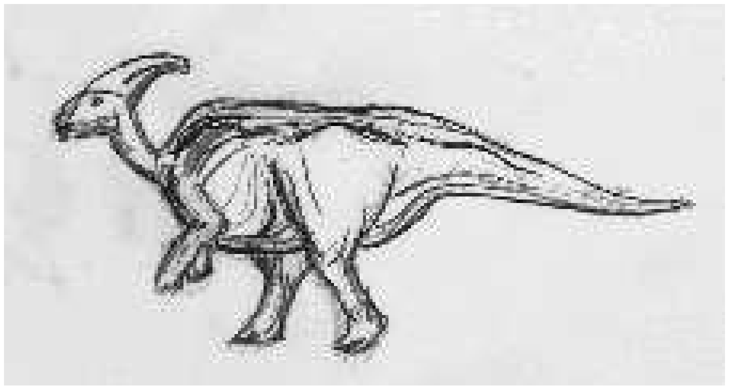


Figure 4.18: Parasaurolophus.

1. certainly **short muzzle and curved hollow horn jutting back from the head**

2. very probably **late cretaceous**

```
1) ! MODULE ORNITHISCHIAN with qc=-0.33, <1> !
2) ! MODULE FOUR_LEGGED_ORNITHISCHIANS with qc=-0.33, <2> !
3) ! MODULE STEGOSAURIDS with qc=-0.27, <3> !*
```

This was the learned path to another group. After backtracking the search is continued[8]:

[8]For complete search see Appendix A.3.3

```
4) ! MODULE ANKYLOSAURS with qc=-0.28, <3> !
5) ! MODULE NODOSAURID_ANKYLOSAURS with qc=-0.26, <4> !*
6) ! MODULE ANKYLOSAURID_ANKYLOSAURS with qc=-0.26, <4> !*

7) ! MODULE SCELIDOSAURIDS with qc=-0.3, <3> !
8) ! MODULE CERATOPSIANS with qc=-0.24, <3> !
9) ! MODULE PSITTACOSAURIDS with qc=-0.21, <4> !
10) ! MODULE PROTOCERATOPSIDS with qc=-0.22, <4> !*
11) ! MODULE SHORT_FRILLED_CERATOPSIDS with qc=-0.23, <4> !*
12) ! MODULE LONG_FRILLED_CERATOPSIDS with qc=-0.24, <4> !*

13) ! MODULE SAUROPODOMORPHS with qc=-0.38, <2> !
14) ! MODULE PROSAUROPODS with qc=-0.29, <3> !
15) ! MODULE PROSAUROPOD_ODDITIES with qc=-0.22, <4> !
16) ! MODULE HERRERASAURIDS with qc=-0.24, <4> !
17) ! MODULE STAURIKOSAURIDS with qc=-0.27, <4> !
18) ! MODULE ANCHISAURIDS with qc=-0.28, <4> !
19) ! MODULE PLATEOSAURIDS with qc=-0.29, <4> !
20) ! MODULE ROCCOSAURIDS with qc=-0.3, <4> !

21) ! MODULE SAUROPODS with qc=-0.29, <3> !
22) ! MODULE DIPLODOCIDS with qc=-0.24, <4> !*
23) ! MODULE CAMARASAURIDS with qc=-0.24, <4> !*
24) ! MODULE CETIOSAURIDS with qc=-0.25, <4> !
25) ! MODULE TITANOSAURIDS with qc=-0.27, <4> !*
26) ! MODULE BRACHIOSAURIDS with qc=-0.27, <4> !

27) ! MODULE ASSORTED_SAUROPODS with qc=-0.41, <3> !*

28) ! MODULE THEROPODS with qc=-0.42, <2> !
..
..
..
55) ! MODULE ORNITHOPODS with qc=-0.34, <3> !
56) ! MODULE THESCELOSAURIDS with qc=-0.27, <4> !*
57) ! MODULE PACHYCEPHALOSAURIDS with qc=-0.26, <4> !*
58) ! MODULE HYPSILOPHODONTIDS with qc=-0.28, <4> !*
59) ! MODULE FABROSAURIDS with qc=-0.29, <4> !
60) ! MODULE HADROSAURIDS with qc=-0.29, <4> !
61) ! MODULE HADROSAURINE_DUCKBILLS with qc=-0.27, <5> !*
62) ! MODULE LAMBEOSAURINE_DUCKBILLS with qc=-0.28, <5> !**
----------------------------------------------
R E S U L T :
```

PARASAUROLOPHUS with qc=-0.16

The recovering is performed by the weakening of the old path and the strengthening of the new path. The next search for species from the group *Lameosaurine duckbills* is done in 16 steps, corresponding to the steps 1..12,28,55,60,62 as shown above. Afterwards the species is found in five steps corresponding to the steps 1,28,55,60,62 as shown above.

4.4.4 Emphasis and forgetting

A medical doctor diagnosing diseases during an influenza epidemic can incorrectly diagnose malaria as influenza. This is because the medical knowledge area corresponding to the influenza disease is primed and the primed area is preferred. Symptoms which are common to both of the two diseases could indicate, to a careless doctor at first glance, a case of influenza. This human behavior of emphasizing knowledge areas which are often used and disregarding knowledge areas which are seldom used is modeled in the next experiment. Mostly this policy is useful, if one has to balance between completeness and speediness of retrieved knowledge.

The taxonomy can be divided into three groups α, β, γ (see tab. 4.19). Each group is represented by nine randomly chosen species (see tab. 4.1). Each species is described by two certain features which are sufficient for the categorization but insufficient to guide the search. The number of required steps S for the categorization is determined (see tab. 4.1). The uniformed search strategy randomly prefers the group γ to group α significantly[9]. After learning the group *Ornitopods* by frequent[10] determination of five species with sufficient knowledge to guide the search with five certain features, the categorization is repeated with equal features as before learning. The number of required steps S^L for the categorization is determined (see tab. 4.1). There is a significant improvement for the groups α, β, and a significant deterioration for the group γ (see tab. 4.2). Emphasis of one knowledge area leads to forgetting of another knowledge area. There is no significant change in the behavior of the whole taxonomy before and after learning, or the groups α and γ together, as they compensate for each other (see tab. 4.2). The significance of the preference of group α to γ is very high[11].

[9] Mean Difference Test rejects the hypothesis that the mean difference is zero between α and γ with p=0.0033.

[10] frequency=4,$\epsilon = 0.1$

[11] Mean Difference Test rejects the hypothesis that hypothesis that the mean difference is zero is even more significant, p = 7.33^{-10}.

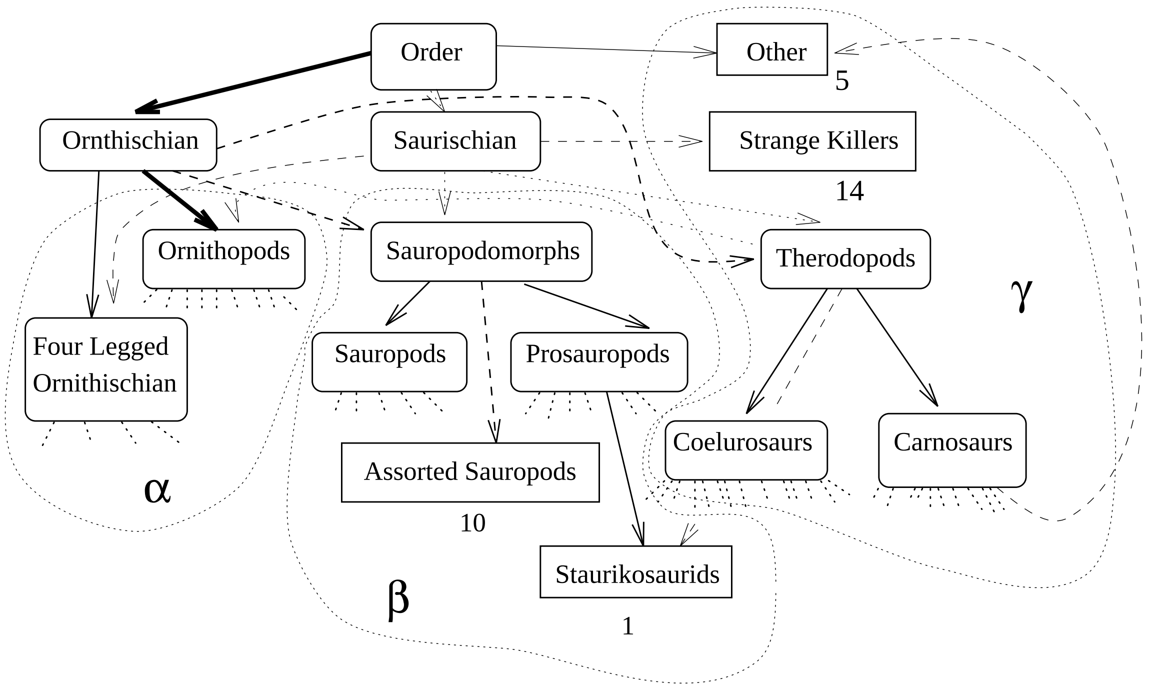


Figure 4.19: Taxonomy of dinosauria after learning Ornithopods.

4.5 Conclusion

A neural model for a deduction system based on the assembly theory was introduced. It was shown that hierarchical categorization can be performed by neural networks efficiently. The categorical representation offers an alternative to the traditional uncertainty calculus [38, 111, 171, 175, 231].

Similarity is used when uncertain knowledge is represented without the need of an additional calculus. In addition, belief tables allow additionally the detachment of the uncertainty of the coded knowledge and of the actually present knowledge. The availability heuristic offers a combination of the frequency with the actual likelihood of the presence of a category. Priming eases the formation of the final hypothesis, as more exact, possible hypotheses are formed. This human behavior of emphasizis of knowledge areas which are often used and disregarding of knowledge areas which are seldom used was modeled. This policy is useful, if one has to balance between completeness and speediness of retrieved knowledge.

The deduction system is a simple model. It is not practical therefore to use it for planning, as planning is mostly described by fewer rules, but which characterizes a much bigger problem space. Planning can be performed more easily by reaction systems in which the premise specifies the conditions that have to be satisfied so that the condition which specifies an action may be undertaken.

SUBORDER	FAMILY	SPECIES	S	S^L
ORNITHOPODS	FABROSAURIDS	**AZENDOHSAURUS**	33	5
	HETERODONTOSAURIDS	**ABRICTOSAURUS**	38	6
	HYPSILOPHODONTIDS	**ALOCODON**	32	4
(α)	TROOEDONTIDS	**TROOEDON**	39	3
	THESCELOSAURIDS	**THESCELOSAURUS**	30	3
	LAMB. DUCKBILLS	HYPACROSAURUS	36	11
FOUR LEGGED	SCELIDOSAURIDS	SCELIDOSAURUS	67	24
	PROTOCERATOPSIDS	LEPTOCERATOPS	60	17
	ANKYLO. ANKYLOSAURS	AMTOSAURUS	66	23
SAUROPODS	CETIOSAURIDS	AUSTROSAURUS	56	36
	TITANOSAURIDS	ALAMOSAURUS	53	37
(β)	DIPLODOCIDS	BAROSAURUS	50	34
	BRACHIOSAURIDS	ASTRODON	54	38
PROSUAROPODS	ANCHISAURIDS	ANCHISAURUS	46	30
	ROCCOSAURIDS	RIOJASAURUS	48	32
	STAURIKOSAURIDS	STAURIKOSAURIDS	41	29
	PROSAUR. ODDITIES	MUSSAURUS	44	27
	HERRERASAURIDS	HERRERASAURUS	45	28
OTHER		MARINE LIZARD	27	65
STRANGE KILL.		MARSHOSAURUS	56	68
COELUSAURUS	COELURIDS	MICROVENATOR	14	52
	NOA. SHANSHANOSAURIDS	NOASAURUS	15	53
(γ)	SEGISAURIDS	SEGISAURUS	6	44
	AVIMIMIDS	AVIMIMUS	17	55
CARNOSAURUS	MEGALOSAURIDS	DILOPHOSAURUS	24	62
	ALLOSAURIDS	ALLOSAURUS	25	63
	TYRANNOSAURIDS	ALBERTOSAURUS	23	61

Table 4.1: Required steps before S and after learning S^LOrnithopods.

Dynamical representation of the problem space is suitable instead of static for reaction systems. No traces can be left behind if problem space representation is used dynamically, because the whole solution sequences must be saved to allow the usage of the availability heuristic.

	S	S^L
mean	44.56	10.67
sdev	15.26	8.56
p	-	4.32^{-7}

steps for α

	S	S^L
mean	48.56	32.33
sdev	5.05	4.09
p	-	5.24^{-9}

steps for β

	S	S^L
mean	23	58.11
sdev	14.05	7.62
p	-	9.73^{-7}

steps for γ

	S	S^L
mean	46.56	21.5
sdev	11.19	12.91
p	-	6.9^{-9}

steps for $\alpha + \beta$

	S	S^L
mean	38.7	33.7
sdev	16.45	20.88
p	-	0.2

steps for $\alpha + \beta + \gamma$

	S	S^L
mean	33.78	34.39
sdev	18.03	25.65
p	-	0.47

steps for $\alpha + \gamma$

Table 4.2: Mean and standard deviation of required steps before and after learning α for α, β and γ and their combinations. The p values were determined by paired sample t test. Significant for $p < 0.05$ by convention.

Chapter 5

The Associative Computer

5.1 Reaction systems

Problems with side effects of actions like planning can be resolved by reaction systems [229]. Reaction systems are a subgroup of production systems. The premise specifies the conditions that must be true before the action described in the conclusion can be taken. The premise and conclusion can be represented by operators. Reaction systems need strategies for conflict resolution. Conflict resolution strategies are often specified by general provisions [76]:

- a rule should be not allowed to fire more than once on the same data
- rules that have used more recent data are prefered
- rules that have a greater number of patterns in the premise are prefered

Rules also can be evaluated by a heuristic function. There are two diferent kinds of heuristic functions:

- The probability that the function is on the best path (see availability heuristic)
- The distance or difference between a given state and the desired state

It is difficult to define heuristic functions, through frequently features can be picked out which describe the distance to the goal [166]. The other possibility is the reusage of solutions to solved problems to indicate which rule to use.

Analogical Problem solving Case-based reasoning [64, 86, 217, 170, 76] is considered a form of analogical problem solving in which specific knowledge of previously experienced cases is used. A new problem is solved by finding a similar past case, and reusing its solution in the new problem.

RETE To generate the conflict set all possible instances are determined by defining all possible sets of working-memory elements, along with all instances of variables. This requires a great deal of time. Often the premises of rules and objects in working memory share common conditions and working memory is only modified a little each time. This observation leads to the "rete" match algorithm [46] which is used by the OPS family of production systems [76]. This algorithm reduces overhead by using a tree-structured sorting network. The patterns in the premise are compiled into this type of network, and the match algorithm computes the conflict set by processing the network. In the network a set of tokens is used for updating working memory changes. The tokens indicate which patterns match which working memory elements, and only this set is updated.

5.2 Problem solvers as parts of bigger systems

5.2.1 Intelligent planning systems

Reaction systems are problem solvers which are very suitable for planning tasks. They are often also the atomic part of more effective planning algorithms. Complex problems are divided into several more-or less independent parts, and these smaller parts are solved seperately by problem solvers. Another possibility is hierarchical abstraction, in which problems are organized into levels of hierarchy. The hierarchy can be used by the problem solvers to make the planning process more effective (for more literature about planing algorithms see. [18, 218, 227, 230]).

5.2.2 The Production system as a model of human problem solving

The SOAR state, operator and result model was developed to explain human problem solving behavior [133]. It is a hierarchical production system in which the conflict-resolution strategy is treated as another problem to be solved. All satisfied instances of rules are executed in parallel in a "temporary" mode. After the temporary execution the best rule is chosen to take action. The decision takes place in the context of a stack of earlier decisions. Those decisions are rated utilizing preferences and added to the stack by chosen rules. Preferences are determined together with the rules by an observer using knowledge about a problem.

5.3 Neural reaction system

The states correspond to pictures which are represented by cognitive entities. The transitions between the states are described by associations [1] and these associations correspond to operators in a symbolic reaction system. These assocations are stored in the permutation associative memory. Working memory is initialized with the initial state description. This description forms a vector $\vec{x}$ which is presented to the permutation associative memory. The permutation associative memory then identifies the set of all appropriate answer vectors $\vec{y^i},\ i \in \{1, \ldots, s\}$ to the question vector $\vec{x}$. Together with the question vector, the answer vectors represent all possible valid associations. In parallel, the answer vectors are executed in a "temporary" mode. After the temporary execution an association which takes action on the working memory is chosen, either randomly, according to a heuristic, or by reusing the solution of already solved problems. This cycle is repeated on the modified working memory until a desired state is reached. By backtracking and the exclusion of loops, a search from the initial state to the desired state is executed. This neural reaction system is called the associative computer and can be integrated into bigger systems.

5.3.1 Permutation associative memory

Permutations

A state is represented by Δ cognitive entities. Associations represent transitions between the states representing pictures. The premise of an association is represented by δ cognitive entities which describe a correlation of objects which should be present. If present, they are replaced by δ cognitive entities of the conclusion. Generally the premise is describe by fewer cognitive entities then the state, $\delta \leq \Delta$. In the recognition phase, all posible δ-permutation of Δ cognitive entities should be composed to test if the premise of an association is valid.

$$\Xi := P(\Delta, \delta) = \frac{\Delta!}{(\Delta - \delta)!}$$

This is done because the premise can describe any correlation between the cognitive entities. Associations can be learned by the associative memory (see fig. 5.4). In the retrieval phase Ξ permutations are formed (see fig. 5.1). Each permutation represents a question vector $\vec{x_i}\ , i \in \{1, \ldots, \Xi\}$. To each question vector $\vec{x_i}$ an answer vector $\vec{y_i}$ with the quality criterion is determined. If the qc value of this answer vector is above a certain threshold, the association can be executed. A copy of the state representation is formed and the corresponding cognitive entities are replaced by the conclusion pattern (see fig. 5.2). Either Ξ

[1] See section 3.4.4.

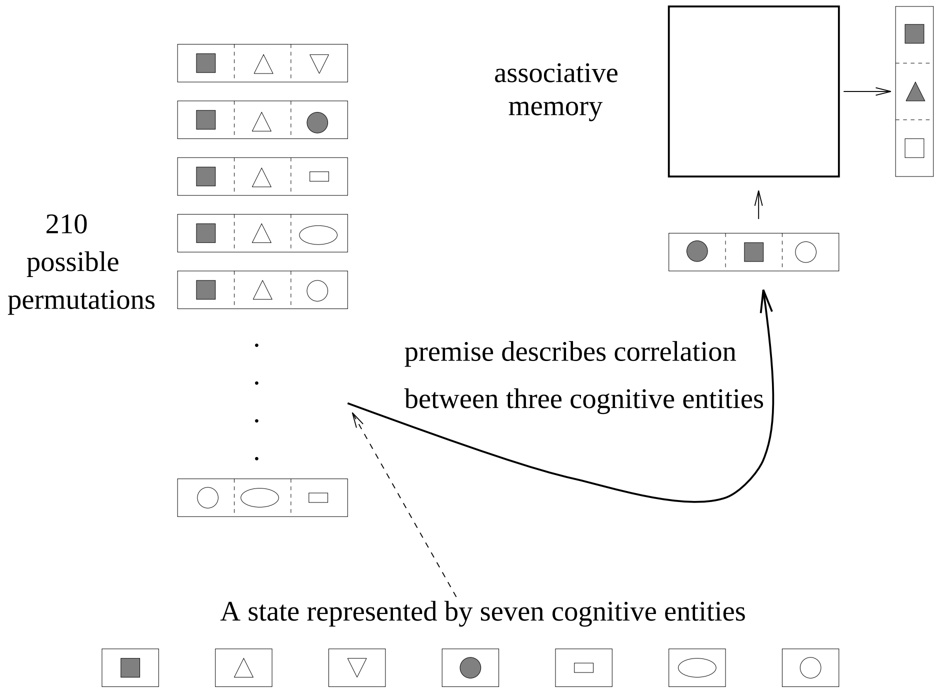

associative memory
210 possible permutations
premise describes correlation between three cognitive entities
A state represented by seven cognitive entities

Figure 5.1: In the recognition phase, all posible 3-permutation of 7 cognitive entities should be composed to test if the premise of an association is valid. (Contents of the cognitive entities are represented for simplicity by geometrical objects without division in associative fields.)

calls of the associative memory are performed, or Ξ copies of associative memory perform the computation in parallel.

Parts

To each cognitive entity in the question vector, a corresponding part of the associative memory is assigned (see fig. 5.3). If we permute the δ arrangement of the cognitive entities we get the same answer vector as before the permutation, but only if the δ parts of the associative memory are permuted accordingly (see fig. 5.3). The parts of the associative memory can be ordered, so that either $\Re$ calls of the associative memory are performed, or $\Re$ copies of the associative memory perform computation in parallel.

$$for\ \delta > 1$$

$$\Re = \frac{1}{\delta} \cdot \sum_{i=1}^{\delta} \frac{\Delta!}{(\Delta - i)!} < \Xi.$$

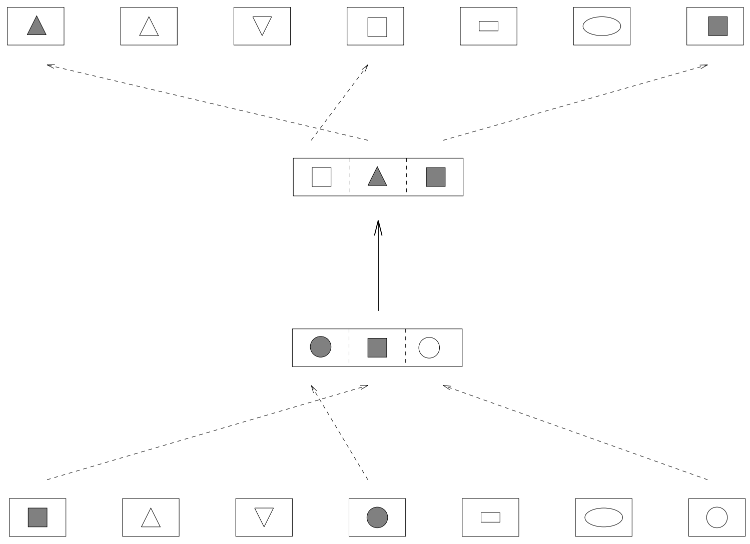


Figure 5.2: A copy of the state representation is formed and the corresponding cognitive entities are replaced by the conclusion pattern.

proof A savings results from the fact that the Ξ permutations are formed from different cognitive entities. The first part of the associative memory computes the output of the Δ cognitive entities. For each Δ first parts there are $\Delta - 1$ different second parts, in total $\Delta \cdot (\Delta - 1)$ combinations. The number of copies of a part can be defined recursively:

$$\#part(1) = \Delta$$

$$\#part(i+1) = \#part(i) \cdot (\Delta - 1).$$

Summed there are:

$$\Re \cdot \delta = \sum_{i=1}^{\delta} \frac{\Delta!}{(\Delta - i)!}$$

parts and

$$\Re \cdot \delta \leq \Xi \cdot \delta$$

applies for parts.

$$for\ \delta > 1; \quad \Re < \Xi.$$

Constraints

Only a small fraction of the Ξ possible answers represent associations. The complexity can be reduced considerably by the fact that the parts of associative memory are also themselves associative memories. The entire associative memory can be composed only from those parts whose *qc* values of answers are above

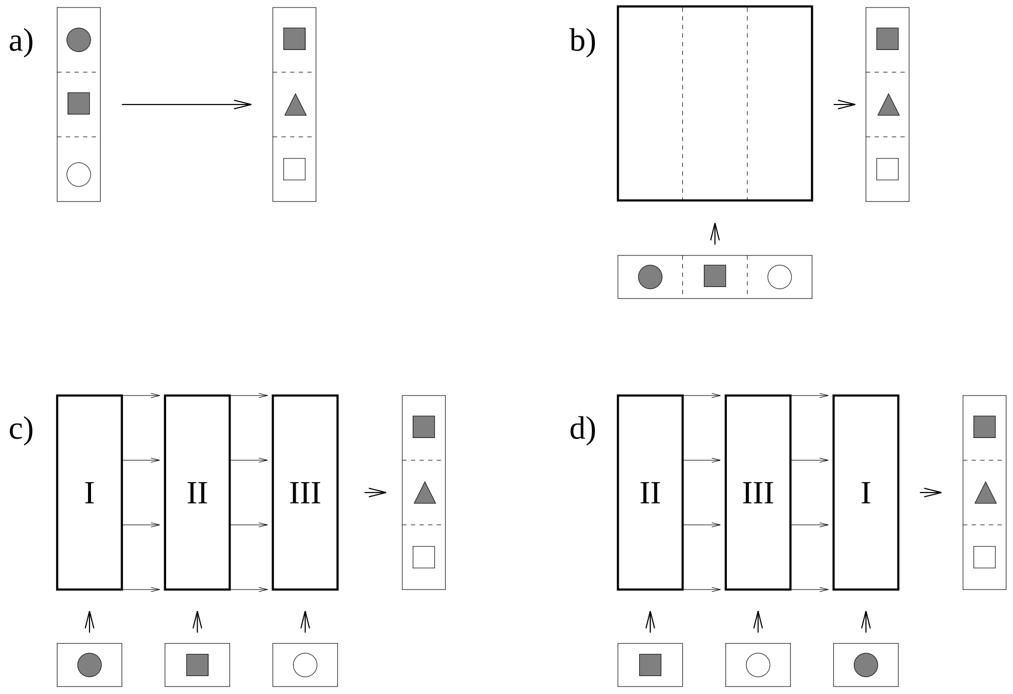


Figure 5.3: a) Association composed of three cognitive entities. b) Recall of the association by the associative memory. c) To each cognitive entity in the question vector, a corresponding part of the associative memory is assigned. d) If we permute the three arrangements of the cognitive entities we get the same answer vector as before the permutation, but only if the three parts of the associative memory are permuted accordingly.

a certain threshold. By this constraint, the number of possible combinations of possible associative memories is reduced.

1. For all parts $\forall i,\ i \in \{1,\dots,\delta\}$ and for all cognitive entities $\forall j,\ j \in \{1,\dots,\Delta\}$ answers and qc_{part} values of $part(i)_j$ are determined ($\delta \cdot \Delta$ times).

2. The qc_{part} values of $\Delta \cdot \delta$ $part(i)_j$ which are over a certain $threshold_{part}$ are marked.

3. From different marked parts whose answers were determined by different cognitive entities associative memories are formed.

4. The corresponding answer vectors with qc_{whole} values of those associative memories are determined.

5. If the qc_{whole} value is over a certain $threshold_{whole}$ an association can be executed.

Architecture After the learning of associations (see fig. 5.4) the retrieval phase occurs. The retrieval phase is subdivided into two stages: the attention

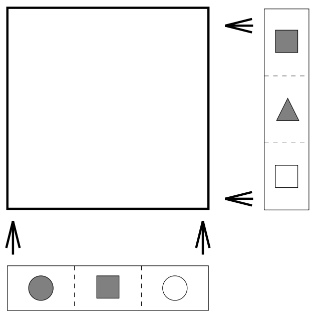


Figure 5.4: Learning phase.

stage and the binding stage. In fig. 5.5 we see the arrangement of the parts in δ layers over Δ cognitive entities. This arrangement is used for the retrieval phase. The question vectors of the parts are represented by the connection between them and the corresponding cognitive entities. In the attention stage $\delta \cdot \Delta$ qc values are determined and the corresponding parts are marked (see fig. 5.5). In the binding stage the associative memories are formed successively from marked parts over different cognitive entities. The formed associative memories determine the answer vectors. This architecture is called the permutation associative memory. Either $\wp$ calls of the associative memory are performed, or $\wp$ copies of the associative memory perform the computation in parallel. If $m(i)$ is the number of marked $parts(i)_j$ for $j \in \{1, \ldots, \Delta\}$, then

$$\wp \leq \Delta + \frac{1}{\delta} \cdot \sum_{i=1}^{\delta} m(i)^i.$$

And there are s possible associations.

$$s \leq \prod_{i=1}^{\delta} m(i).$$

notice

$$\forall i,\ i \in \{1, \ldots, \delta\}, m(i) \leq \left\lfloor \left(\frac{\Delta!}{(\Delta - \delta)!} - \frac{\delta}{\delta - 1} \cdot \Delta \right)^{\frac{1}{\delta}} \right\rfloor$$

$$\Longrightarrow \wp < \Re.$$

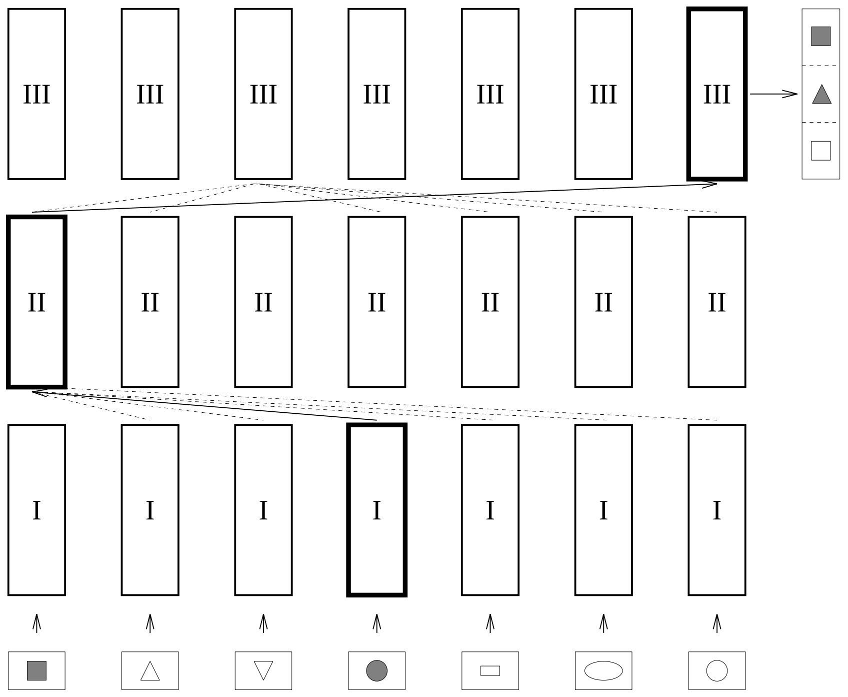


Figure 5.5: The permutation associative memory. In the attention stage parts are marked (thick boxes). In the binding stage the associative memories are formed (thick arrows).

proof The estimation $\forall i,\ i \in \{1,\ldots,\delta\}\ \frac{\Delta!}{(\Delta-i)!} \leq m(i)^i + \Delta$ leads to the trivial solution $\Delta - \Delta = 0 = m(1)$. So we begin the estimation at i=2, $\forall i,\ i \in \{2,\ldots,\delta\}\ \frac{\Delta!}{(\Delta-i)!} - \frac{\delta}{\delta-1}\cdot\Delta \leq m(i)^i$. Because $\forall i,\ i \in \{2,\ldots,\delta\},\ \ \delta < \Delta$ $\left(\frac{\Delta!}{(\Delta-(i+1))!} - \frac{\delta}{\delta-1}\cdot\Delta\right)^{\frac{1}{i+1}} < \left(\frac{\Delta!}{(\Delta-i)!} - \frac{\delta}{\delta-1}\cdot\Delta\right)^{\frac{1}{i}} \Longrightarrow \forall i,\ i \in \{1,\ldots,\delta\}, m(i) \leq \left(\frac{\Delta!}{(\Delta-\delta)!} - \frac{\delta}{\delta-1}\cdot\Delta\right)^{\frac{1}{\delta}} \Longrightarrow \wp < \Re$.

Given a state represented by a unit, the permutation associative memory recognizes s question vectors, s copies of state representation are formed. The cognitive entities which form the question vectors (premise) are replaced by the cognitive entities of the answer vectors (see fig. 5.2). The resulting s states are represented by s units (see fig. 5.6).

Qualities

- Permutation associative memory is an associative memory which allows the representation of several answers corresponding to a single input.
- The retrieval phase of the permutation associative memory corresponds

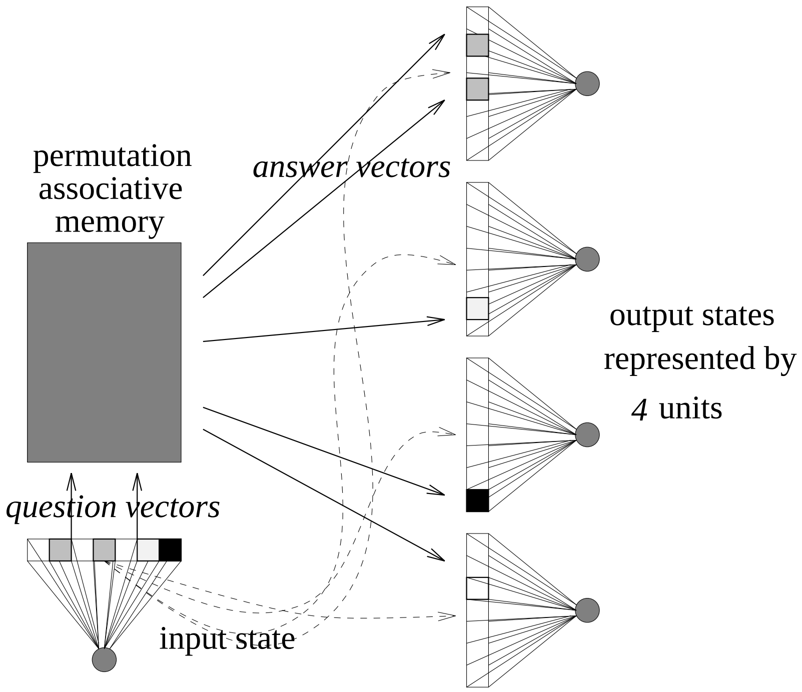

permutation associative memory
answer vectors
output states represented by 4 units
question vectors
input state

Figure 5.6: The cognitive entities which form the question vectors (premise) are replaced by the cognitive entities of the answer vectors (conclusoion). The resulting 4 states are represented by 4 units.

to the determination of all possible instantiations of rules in a symbolic production system.

- The retrieval phase is subdivided into two stages: the attention stage [5, 155] and the binding stage. Only when the attention is focused on certain parts are they bound to a whole object [210, 34, 155] representing a question.
- The process of marking corresponds to the mechanism of attention window mechanisms that selects a pattern in the visual buffer for further aceesses of the visual system in the human brain [88].
- The succesive examination of the cognitive entities corresponds to the spotlight theory [37]. Attention is linked to a spotlight that is focused on the cued location and shifted as necessary [88, 155].
- The binding problem[2] is solved after the attention stage by constraining the formation of permutations of parts of an associative memory.
- The configuration of cognitive entities describing a scene can be arbitrary.

[2]For more information about the binding problem, see section 7.1.2.

Example from the geometric blocks world

In this example, blocks can be placed in three different positions and picked up and set down by a robot arm. There are two different classes of blocks: cubes and pyramids. No other block may be placed on top of a pyramid, while either type of block may be placed on top of a cube. The robot arm is represented in the upper right corner. Three objects are present; two cubes and one pyramid. Eighteen

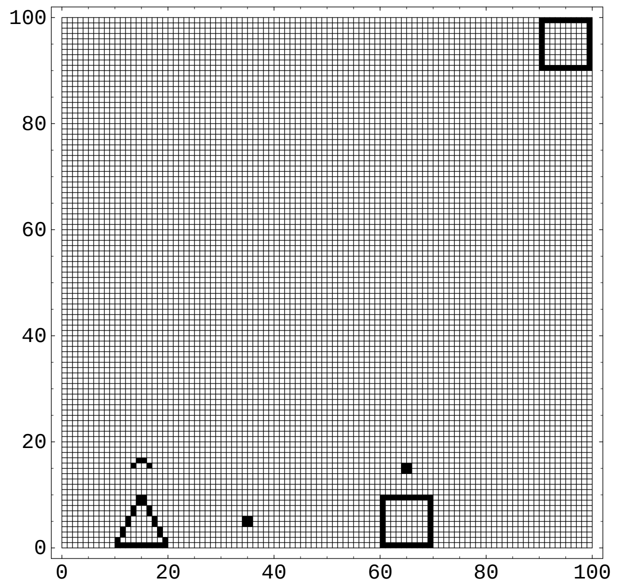


Figure 5.7: The world for the task geometric blocks.

different states are possible. The state of the world is described by a hundred times hundred pixel pictures (see fig 5.7). These pictures are represented by seven cognitive entities. The objects are described by ten times ten pixel associative fields, the position coordinates by two hundred pixel vectors. Thirty associations are learned. Its premises and conditions are described by three cognitive entities (see fig. 3.10 and fig. 3.14). Together the question and the answer vector have the dimension 900 ($100 \cdot 3 \cdot 3$). After learning, a weight matrix of dimension 900^2 emerges (see fig. 5.8). The matrix is symetric since for each assoication there exists an inverse association and vice versa. The matrix is low loaded, but the weights are not equally distributed. They form ordered clusters with nearly fractal properties. The order is reflected in the structure of the weight matrix represented in fig. 5.9. The weight matrix is composed of 10 elementary blocks. The blocks represent nearly a linear correlation with peaks at the blocks with sum values 122 and 224.

In the following example, associations corresponding to the state represented in fig. 5.15-ID4 are determined. The representation of the state ID4 by seven cognitive entities as used for the computer simulations follows:

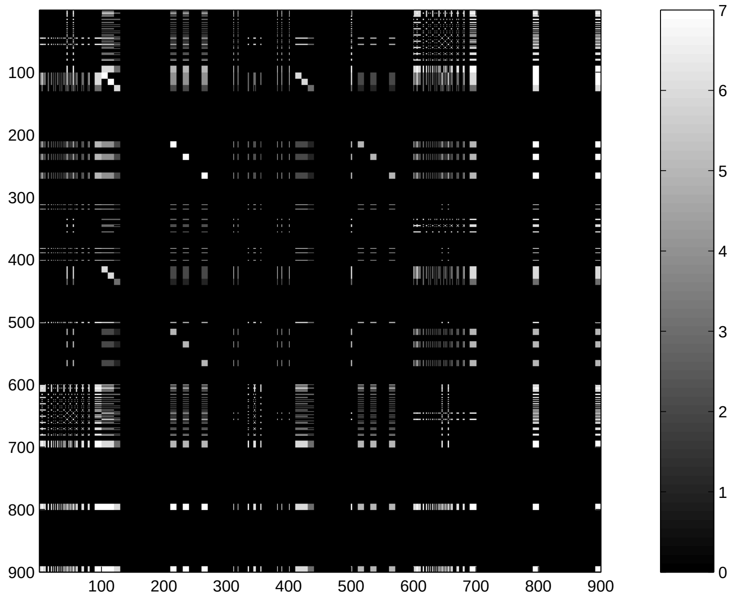


Figure 5.8: The weight matrix of the permutation associative memory for the task geometric blocks. 44448 synapses are not zero, 5.4874% of the weight matrix. The grey level bar on the right indicates the scaling of the frequency weight values.

```
No:1

0000110000
0000110000
0001001000
0001001000
0010000100
0010000100
0100000010
0100000010
1000000001
1111111111

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000


No:2

1111111111
1000000001
1000000001
1000000001
1000000001
1000000001
1000000001
1000000001
1000000001
```

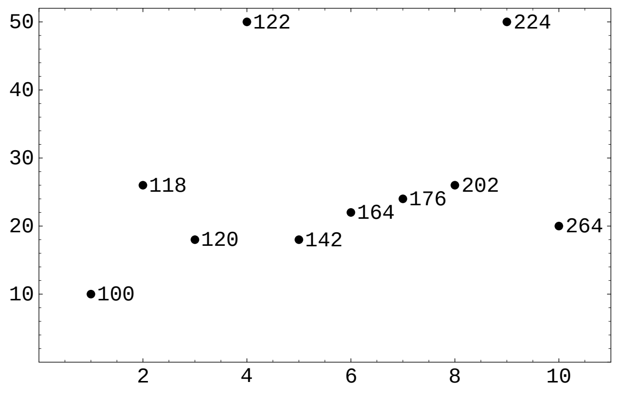


Figure 5.9: The sum of the synapses of a column. The x axis indicates the frequency of different sum values of columns. The written number corresponds to the sum values.

```
1111111111

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000001111111111000000000000000000000000000000000000000000000000000000000000

No:3

1111111111
1000000001
1000000001
1000000001
1000000001
1000000001
1000000001
1000000001
1000000001
1111111111

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000001111111111000000000000000000000000000000

No:4

0000000000
0000000000
0000000000
0000000000
0000110000
0000110000
0000000000
```

```
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000001111111111000000000000000000000000000000

No:5

0000000000
0000000000
0000000000
0000000000
0000110000
0000110000
0000000000
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000001111111111000000000000000000000000000000000000000000000000000000000000


No:6

0000000000
0000000000
0000000000
0000110000
0001001000
0000000000
0000000000
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000


No:7

0000000000
0000000000
0000000000
0000000000
0000110000
0000110000
0000000000
0000000000
0000000000
0000000000

0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111

0000000000000000000000000000000000000000000000000000000000000000000000000000000000000000001111111111
```

Three layers over seven cognitive entities are used. The hard threshold strategy is used with $threshold_{part} = 0.7$ and $threshold_{whole} = 0.87$. $5 + 3 + 1$ parts of associative memory are marked. Three possible answers are recognized corresponding to the gripping of three diferent blocks by the robot arm. $\Xi = 200$ permutations are possible, the estimation for $m(1), m2, m(3)$ *is* ≤ 5. The three parts correspond to the three layers adressed by `Nr.i` are determined during the attention stage. The number after the address of the cognitive entity `No.j` corresponds to the *qc* value of the answer. (Answer of the part `Nr.i`, cognitive entity `No.j` is the question.)

```
Attention stage (threshold is 0.7):

Nr.1

No.1= 1, No.2= 1, No.3= 1, No.4= 1, No.5= 1, No.6= -1, No.7= -1.

Nr.2

No.1= -1, No.2= -1, No.3= -1, No.4= 1, No.5= 1, No.6= 1, No.7= -1.

Nr.3

No.1= -1, No.2= -1, No.3= -1, No.4= -1, No.5= -1, No.6= -1, No.7= 1.
```

Associative memories are formed from three different marked $parts(i)_j$. Notation: i is represents the position, j the number. The qc_{whole} values over the $threshold_{whole}$ are marked by `+`.

```
Binding stage (threshold is 0.87):

1, 4, 7,   has qc=0.372593
1, 5, 7,   has qc=0.372593
1, 6, 7,   has qc=1 +
2, 4, 7,   has qc=0.588889
2, 5, 7,   has qc=1 +
2, 6, 7,   has qc=0.448642
3, 4, 7,   has qc=1 +
3, 5, 7,   has qc=0.588889
3, 6, 7,   has qc=0.448642
4, 5, 7,   has qc=-0.367273
4, 6, 7,   has qc=-0.367273
5, 4, 7,   has qc=-0.367273
5, 6, 7,   has qc=-0.367273
```

Three possible answers are recognized corresponding to the gripping of three diferent blocks by the robot arm.

In (fig. 5.49 c-I4) there is a noisy representation of the state of (fig. 5.15 ID4). Since the blocks are shifted and, in addition, some pixels are deleted and some are added, the hard threshold strategy fails. Therefore, the soft threshold strategy is used. Because of the soft threshold strategy and the noise, more parts are marked. $7 + 3 + 1$ parts of associative memory are marked. Three possible answers are recognized.

```
Attention stage (threshold is 0.7):

Nr.1

No.1= -1, No.2= -1, No.3= -1, No.4= 1, No.5= 1, No.6= -1, No.7= -1.

Nr.2

No.1= -1, No.2= -1, No.3= -1, No.4= 1, No.5= 1, No.6= 1, No.7= -1.

Nr.3

No.1= -1, No.2= -1, No.3= -1, No.4= -1, No.5= -1, No.6= -1, No.7= 1.


Binding stage (threshold is 0.87):

4, 5, 7,   has qc=-0.367273
4, 6, 7,   has qc=-0.367273
5, 4, 7,   has qc=-0.367273
5, 6, 7,   has qc=-0.367273


!THRESHOLD CHANGED FROM HARD TO SOFT!

Attention stage (threshold is 0.7):

Nr.1

No.1= 0.79, No.2= 0.82, No.3= 0.82, No.4= 1, No.5= 1, No.6= 0.99, No.7= 0.93.

Nr.2

No.1= 0.2, No.2= 0.17, No.3= 0.22, No.4= 1, No.5= 1, No.6= 1, No.7= 0.28.

Nr.3

No.1= 0.33, No.2= 0.4, No.3= 0.4, No.4= 0.03, No.5= 0.03, No.6= -1, No.7= 1.
```

```
Binding stage (threshold is 0.87):

1, 4, 7,   has qc=0.28321
1, 5, 7,   has qc=0.28321
1, 6, 7,   has qc=0.876667 +
2, 4, 7,   has qc=0.493333
2, 5, 7,   has qc=0.883077 +
2, 6, 7,   has qc=0.359259
3, 4, 7,   has qc=0.883077 +
3, 5, 7,   has qc=0.493333
3, 6, 7,   has qc=0.359259
4, 5, 7,   has qc=-0.367273
4, 6, 7,   has qc=-0.367273
5, 4, 7,   has qc=-0.367273
5, 6, 7,   has qc=-0.367273
6, 4, 7,   has qc=0.0813008
6, 5, 7,   has qc=0.0813008
```

Again, despite noise, three possible answers are recognized corresponding to the gripping of three diferent blocks by the robot arm.

5.3.2 Search chain

A neural reaction system represents the problem space as a search chain to prohabit the formation of loops and allow the possibility of backtracking. A sequence of states of pictures described by cognitive entities can be represented by connected units (see fig. 3.13 and fig. 5.10). After the temporary parallel execution

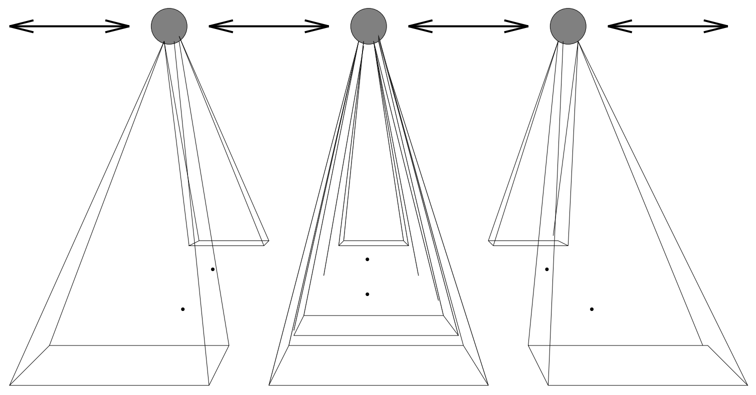


Figure 5.10: A sequence of states of pictures described by cognitive entities can be represented by connected units.

of a chosen state, s new states emerge. The s states are represented by a layer of s units. From the s^t states, one state is chosen and the new s^{t+1} states are determined. They are represented by another layer of s^{t+1} units (see fig. 5.11). The chosen units are marked and represent the sequence of carried out states.

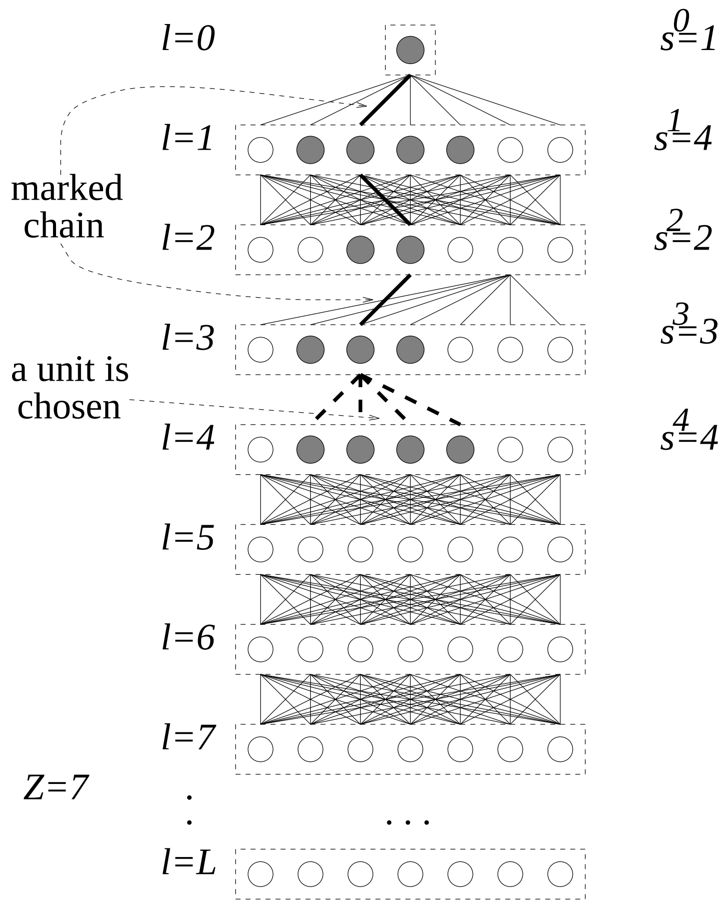


Figure 5.11: A search chain of units. (Cognitive entities represented by units are not illustrated.)

The search chain consits of L layers of each z units. $\forall t,\ s^t \leq z$. The start of the search chain consists of one unit representing the initial state, the layer $l = 0$. This unit is connected with all units of the first layer, $l = 1$. The units of each layer are connected with all units of the following layer (see fig. 5.11). These connections correspond to edges in a search tree. The sequence of carried out states forms a marked chain. By comparison with those states, loops can be prevented and a search can be realized. A state can cause an impasse when no valid transition to a succeeding state exists. In this case, the marked chain allows backtracking to the previous state. Another state can be chosen, if possible, or backtracking is repeated. The number of connections between the marked units corresponds to the depth of the search tree. The resulting search strategy is the "deep search" strategy. An example of deep search strategy performed by the search chain is shown (fig. 5.12). A search chain corresponds to the stack used in the symbolic reaction systems.

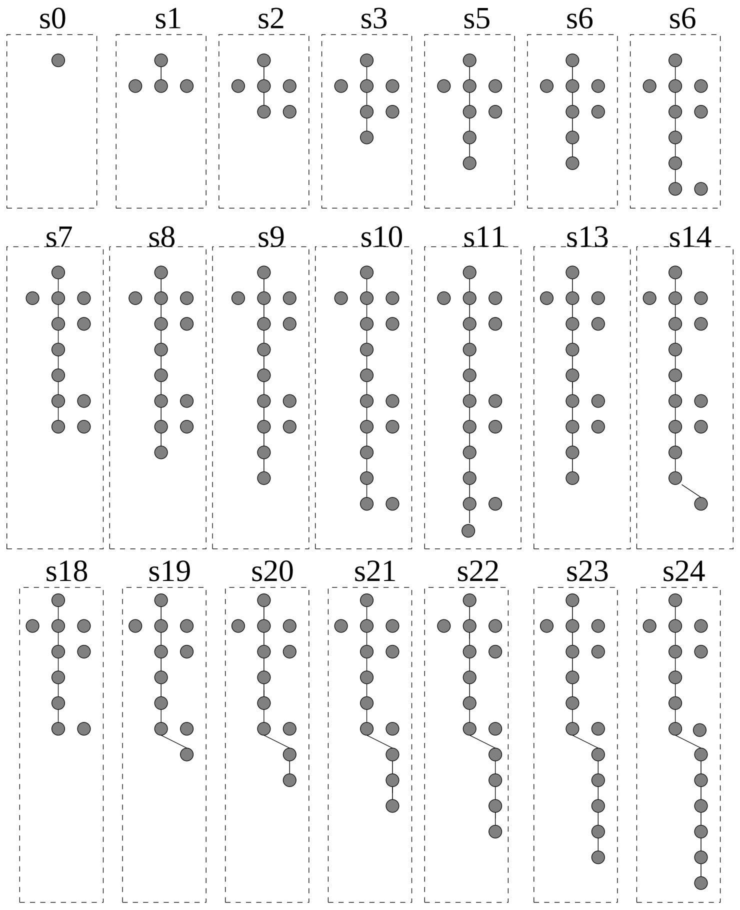


Figure 5.12: The deep search strategy performed by the search chain. (Compare fig. 5.17 amd 5.18.)

Chains in cortex

Stimuli are transmited by chains of neurons in the cortex [1, 219]. Stimuli can be transmitted along such chains in two ways: asynchronously and synchronously [1]. In the asynchronous mode the excitation goes from one neuron of the chain to the next. In the synchronous mode some cells of the chain fire at the same time. The synchronous mode results from a synfire chain architecture [1]. The synchrous mode could represent the sequence of carried out states, the asynchrous, the propagation of search.

5.3.3 Controller

The search can be described by the following algorithm:

1. The computation begins at the unit representing the initial state, layer $l = 0$.
   (a) The unit is marked.
   (b) All resulting possible states are computed by the permutation associative memory and are stored in the layer $l = 1$.
   (c) If there are no answers found by the permutation associative memory the computation fails.
2. *Test*, two tests are performed on all states represented by the layer l.
   (a) First it is determined if the desired state was reached. If yes, the computation halts successfully.
   (b) Second, it is determined if the state already occured in the marked chain. If yes, the state is disabled to prohibit the formation of loops.
3. *Chose* a unit of a layer l if posssible.
   (a) If all units of a layer l are disabled:
      - Units of this layer l are enabled for further usage.
      - The marked unit of the layer $l - 1$ is disabled.
      - Backtracking to the previous layer is done, $l = l - 1$
      - If backtracking to the initial state occurs $l = 0$, the computation fails.

   (b) From the units of the layer l one unit is chosen[3].
4. Computation of a new state for a chosen unit of layer l.
   (a) All resulting possible states are computed by the permutation associative memory and are stored in the layer $l = l + 1$.
   (b) If no answer for this state exists:
      - The corresponding unit of layer l is disabled.
      - Goto *Chose* .

   (c) If $l > L$, computation fails because of lack of resources, L number of layers.
   (d) The chosen unit is marked.

[3]The first unit of a layer is always chosen. Conflict resolution stratagy can be used here.

5. Goto *Test*.

Besides the permutation associative memory and the search chain, two other building blocks are needed to allow the realization of the search algorithm.

A pattern matcher computes the $qc_{Ca}(b)$ of a state b represented as a picture in regard to category Ca, also represented as a picture.
For all $b(i)$ $i \in \{1,..,s\}$ states represented in a layer and in the category coreponding desired state, the $qc_{Ca}(b(i))$ is computed in the first test phase. If the $qc_{Ca}(b(i))$ value is 1 the desired state is reached.
In the second phase for all $b(i)$ $i \in \{1,..,s\}$ states represented in a layer and for all Ca states in the marked chain, $qc_{Ca}(b(i))$ values are computed sequentially. The corresponding states $b(i)$ which occur in the marked chain are disabled. In case Ca is a marked state, $qc_{Ca}(b(i))$ is 1.

Backtracker enables units of a layer l if all units are disabled. The marked unit of the layer $l-1$ is disabled, and the active layer is the layer $l = l-1$.

Together the moduls are linked by a controler which can be described by a deterministic finit automata. Deterministic automaton can be described by neural networks (see [219]). Because the activation is propagated from unit to unit by connections, local computation is performed as postulated by biology. The control results from the diverse behavior of permutation associative memory, pattern matcher, and backtracker.

Permutative Associative memory's behavior is:

1. *Compute* To a chosen unit of layer l all resulting possible states are computed and are stored in the layer $l+1$.

2. *No answer* The chosen unit of the layer l is disabled if no answers exist.

3. *Fail* The computation fails if no answer for the initial state exists, $l = 0$

Pattern matcher behavior is:

1. *Desired state* If a desired state is reached by one of the units of the layer l the computation succeeds.

2. *Chose* From the units of a layer l one unit is chosen. The first unit of the layer is allways chosen.

3. *No loops* Disables units of the layer l which cause loops by retracing of the marked chain (see fig.).

Backtracker behavior is:

1. *Backtrack* If all units of a layer l are disabled, enable all units of a layer l. Disable the marked unit of the layer $l-1$. Activate layer $l = l-1$.

2. *Fail Backtrack* If all units of a layer $l = 1$ are disabled.

Deterministic finite automaton

A deterministic finite automaton [107] is a quintuple $M = (K, \Sigma, \sigma, s, F)$ where K is a finite set of states, Σ is an alphabet, $s \in K$ is the initial state, $F \subseteq K$ the set of final states, and δ, the transfer function, is a function from $K \times \Sigma$ to K. If the automaton is in state $q \in K$ and the symbol is $\sigma \in \Sigma$, then $\delta(q, \sigma) \in K$ is the uniquely determined state to which the automaton passes. The symbol set corresponds in the controller to the behavior of the modules. An additional behavior is the marking of the chosen units. The controller is described by the deterministic finite automaton (see fig. 5.13):

$$K = \{q_0, q_1, q_2, q_3, q_4, q_5, f_0, f_1, Goal\}$$

$$\Sigma = \{mark, fail, compute, desired\ state, no\ loops, backtrack,$$
$$fail\ backtrack, chose, no\ answer\}$$
$$s = \{q_0\}$$
$$F = \{f_0, f_1, Goal\}$$

q	σ	$\delta(q,\sigma)$
q_0	mark	q_1
q_1	fail	f_0
q_1	compute	q_2
q_2	desired state	*Goal*
q_2	no loops	q_3
q_3	backtrack	q_3
q_3	fail backtrack	f_1
q_3	chose	q_4
q_4	no answer	q_3
q_4	compute	q_5
q_5	mark	q_2

5.3.4 The Associative Computer

The associative computer is composed of permutation associative memory in which the associations are stored. It also contains a search chain which dynamicaly represents the search space. The controller links the permutation associative

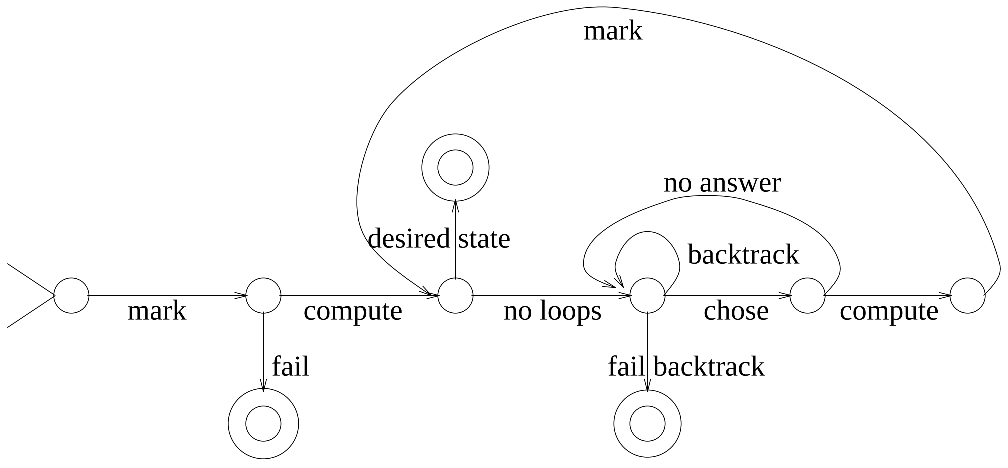


Figure 5.13: The controller is described by the deterministic finite automaton.

memory and the search chain and controls the computation (see fig. 5.14). The associative computation is specified by the stored associations and by the initial and desired state. Fig. 5.15 represents examples of initial and desired states for the geometric block world as pictures.

Fig. 5.16 represents the associative computation ID4 $\rightarrow$ ID5 without backtracking. The depth-first search led to no impasse. Ten steps were needed until the desired state was reached, because ten associations were performed. The plan length is ten. The search sequence is represented by pictures coresponding to the representation of the states of the world by the seven cognitive entities. The initial and desired states are included. Fig. 5.17 and fig. 5.18 represent the associative computation ID4 $\rightarrow$ ID1. The solution of the problem is represented by twelve assocations. The plan length is twelve. The solution was found in twenty-four steps. The steps are abbreviated by s in the fig.5.17, 5.18 , the depth of the search tree by d. Six backtracking steps were performed. The first two backtracking steps occur at step 11 depth 11 backtracking to the previous step 9 depth 9. The corresponding state is listed again for clarity reasons. It becomes step 13 depth 9 and leads to step 14 depth 10. This state is a impasse, and therefore four backtracking steps are performed to the previous step 6 depth 6. It becomes step 18. From this the step continues until the desired state is reached without backtracking.

It was shown how a reaction system can be implemented by neural networks. Not only ideas on such a system were presented, but also computer simulations of implemented systems doing computations. The system, the associative computer, has the same behavior as a purely symbolical reaction system (production system). It was shown how a mapping of a symbolical reaction system to a neural network can be performed. For a cognitive-science scientist it is important to un-

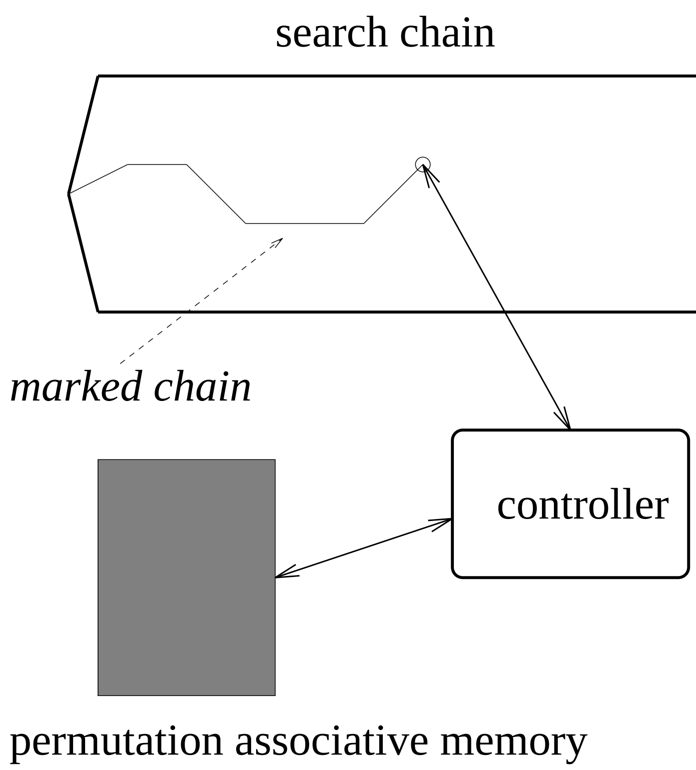


Figure 5.14: The controller links the permutation associative memory and the search chain and controls the computation.

derstand how such computations can be performed by the human brain. Another question is, whether neural systems increase our understanding of cognition or not. Are there advantages of performing this computation by neural networks or not? An important research field in Artificial Intelligence is the usage of heuristics to speed up the search. It is not easy to define heuristic functions, as there is no rule which says how to do this. A heuristic function results automatically in the associative computer from the representation of knowledge by pictures. Another approach to the view of heuristic functions is shown: heuristic functions which result from the manner of description of the knowledge. Other properties of the associative computer are robustness and the possibility of learning with a teacher or from experience. These properties are examined in the next section.[4]

[4]Appendix A.2 illustrates the organization of the object oriented laboratory and describes information concerning the implementation of the associative computer.

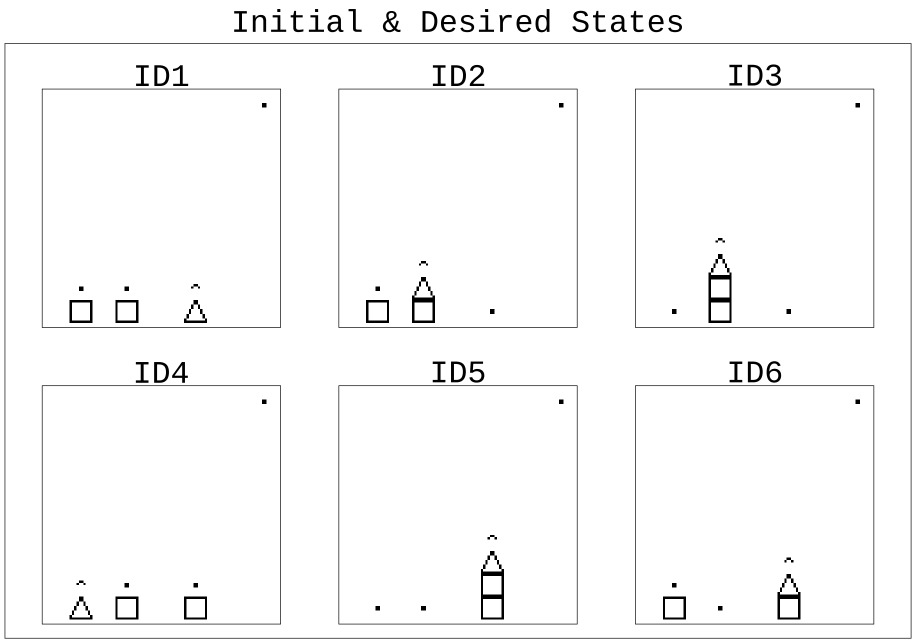


Figure 5.15: The initial and desired states for the task geometric blocks.

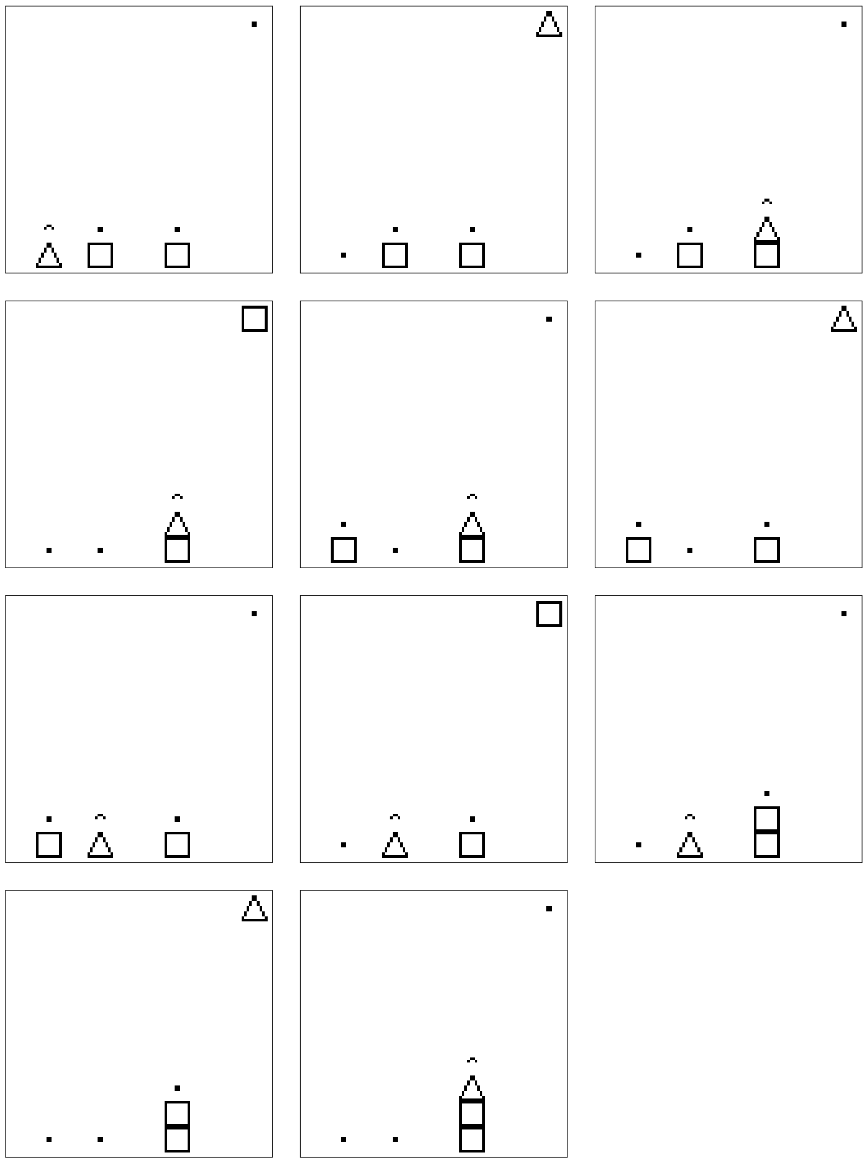


Figure 5.16: Planning of the task ID4 → ID5 without heuristic. The depth-first search led to no impasse. Ten steps were needed until the desired state was reached, because ten associations were performed. The plan length is ten. The search sequence is represented by pictures coresponding to the representation of the states of the world by the seven cognitive entities. The initial and desired states are included.

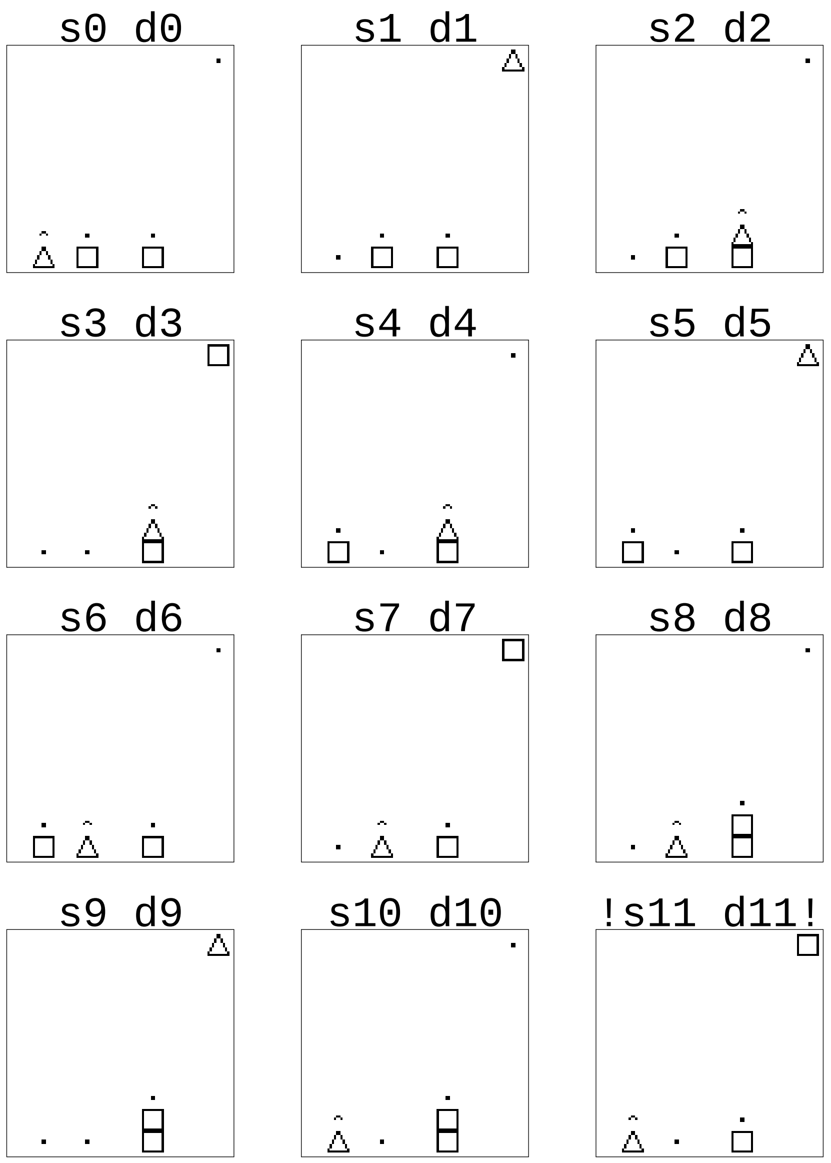


Figure 5.17: Planning of the task ID4 → ID1 without heuristic until the first impasse. The solution of the problem is represented by twelve assocations. The plan length is twelve. The solution was found in twenty-four steps. The steps are abbreviated by *s* (continued in the fig. 5.18), the depth of the search tree by *d*.

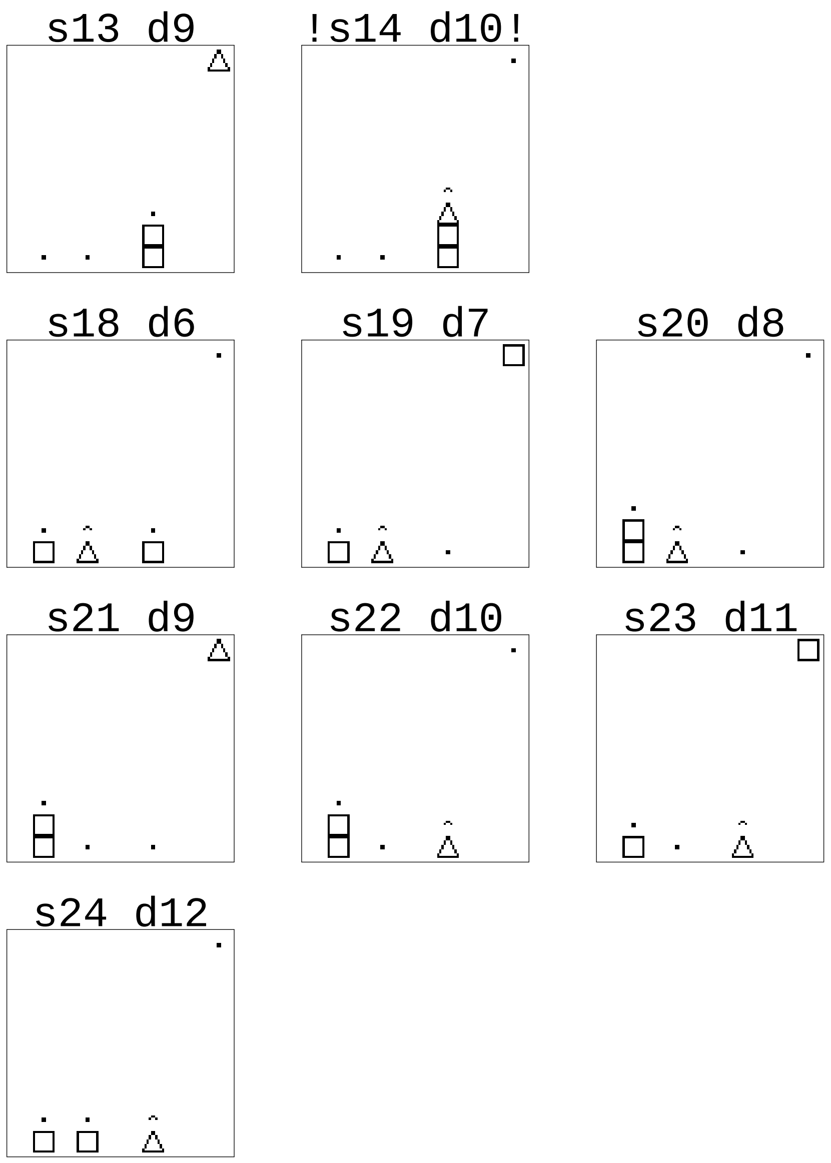


Figure 5.18: Planning of the task ID4 → ID1 without heuristic. The first two backtracking steps occur at step 11 depth 11 backtracking to the previous step 9 depth 9. The corresponding state is listed again for clarity reasons. It becomes step 13 depth 9 and leads to step 14 depth 10. This state is a impasse, and therefore four backtracking steps are performed to the previous step 6 depth 6. It becomes step 18.

5.3.5 Pattern heuristics

The $qc_{Ca}(b)$ value computed by the pattern matcher can be interpreted as a heuristic function (see section 3.3.2). The pattern matcher behavior (point 2) for chosing a unit of a layer l is changed to:

- For all $b(i)$ $i \in \{1, .., s\}$ states represented in a layer, the unit whose state has the greatest $qc_{desired\ state}(b(i))$ value is chosen.(Point 3.b in the description of the algorithm.)

The $qc_{desired\ state}$ is computed using the vectors of dimension $100 \cdot 100$ resulting from picture representation. The resulting search strategy is the hill climbing search strategy. This strategy is analogous to hierarchical categorization with the difference that instead of the probability that the function is on the best path, the distance to the desired state is given. This heuristic function is called *h1*. Blind-search strategy without any heuristic function is called *h0* (see fig 5.17). The *h1* value appears not to be practical as a heuristic function in the block world, because no applicable information is given for states corresponding to the robot arm holding a block. This state occurs alternately with other states in every planning sequence in a block world. The $qc^2_{Ca}(b)$ function is introduced to eleminate the partial blindness in the block world of $h1$. If a state corresponds to the robot arm holding a block, a prediction is made. The state receives the maximal $qc_{desired\ state}$ value of all resulting possible states. For this task the possible states are computed temporarily by the permutation associative memory (see fig. 5.19). This heuristic function which eliminates the sightlessnes in the block world is called *h2*. The pattern matcher behavior for chosing a unit of a

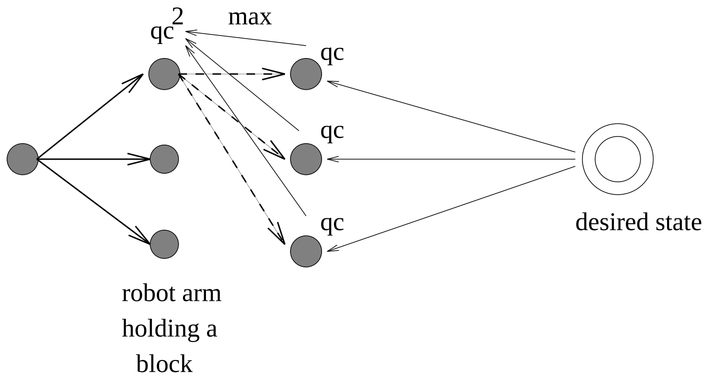


Figure 5.19: If a state corresponds to the robot arm holding a block, a prediction is made. The state recieves the maximal $qc_{desired\ state}$ value of all resulting possible states. For this task the possible states are computed temporarily.

layer l is changed to:

- For all $b(i)$ $i \in \{1, .., s\}$ states represented in a layer, the unit whose state has the greatest $qc^2_{desired\ state}(b(i))$ value is chosen.

The *h2* heuristic function brings significant improvment in comparison to a blind-search *h0* and the part-blind *h1* function in the block world, see fig. 5.20, tab. 5.1.

	h0	h1	h2
mean	14.4	15.53	10.47
sdev	10.20	11.05	7.25
p	-	0.33	**0.0089**

Table 5.1: Mean and standard deviation value of the required steps for all 30 combinations of initial and desired states. The p values were determined by the paired sample t test. Significant for $p < 0.05$ by convention.

	h0	h1	h2
mean	11.07	11.8	8.93
sdev	6.23	6.26	5.5

Table 5.2: Corresponding mean and standard deviation value of required plan length for all 30 combinations.

	h0	h1	h2
mean	1.67	1.87	0.77
sdev	3.33	2.91	1.5

Table 5.3: Corresponding mean and standard deviation value of required back-tracking steps for all 30 combinations.

The first intuitive idea, that a state represented as a picture is nearer to a desired state represented as a picture, the more similar those pictures are, was statistically confirmed. In fact, a pattern heuristic speeds up the search. The real world give us insights on how to solve the problem. Such an insight is used by the pictorial representation and the resulting similarity criterion. This information is not always correct, but it is generally better to obey it than to ignore it.

5.3.6 Prediction heuristic

The solved problems are reused to speed up the search for related or similar problems. Due to the state description by pictures, a prediction heuristic can

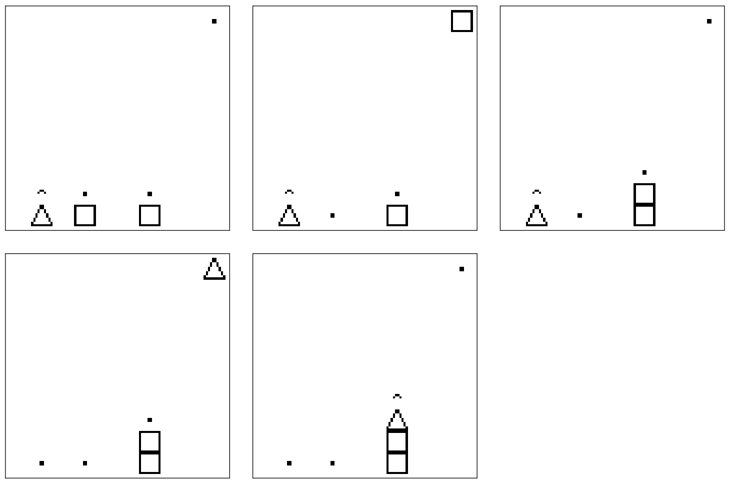


Figure 5.20: Planning of the task ID4 → ID5 with h2 heuristic function. Compare fig. 5.16.

be defined. It gives a probable value that the function is on the best path. A succeeding state in a reaction system depends only on the current and the desired state. Given a current state, together with knowledge of the desired state, the next state can be predicted. The current state together with the desired state represent the question, the next state the answer. A question vector is composed by concatenation of the pattern representation of the current state and the desired state. The answer vector coresponds to the pattern representation of the next state. Both vectors can be stored in the associative memory. Vectors resulting from the pattern representation are preferred to the vectors resulting from the ordered cognitive entity representation because they are more sparse. A resulting sequence of states describing a plan is stored in an associative memory. The sequence commences with the initial state and ends with the penultimaste state before the desired state. From each current state together with the desired state a question vector is formed and stored together with the answer vector coresponding to the next state. After "learning" the sequence can be recalled. By posing the question vector which is composed of the initial state, holding the desired state part, and by feedback, the answer vector to the question vector, the state sequence is determined ([143] see fig. 5.21). This method, however, is unre-

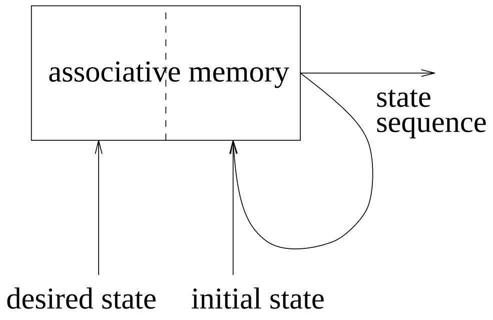


Figure 5.21: By posing the question vector which is composed of the initial state, holding the desired state part, and by feedback, the answer vector to the question vector, the state sequence is determined [143].

liable, and because of superimpositions and overloading, errors can occur. More suitable is the realization of a prediction heuristic in an associative computer with the aid of an associative memory. This associative memory is called the prediction associative memory, and becomes another building block of the associative computer. The permutation associative memory determines $b(i)$ states of a chosen state. At the same time, the prediction associative memory determines with the soft threshold strategy the answer pattern $Pred$ to this chosen state together with the desired state. The unit with the greatest $qc_{Pred}(b(i))$ value is chosen (see fig. 5.22). The pattern matcher behavior for chosing a unit of a layer l is changed to:

- For all $b(i)$ $i \in \{1, .., s\}$ states represented in a layer, the unit whose state has the greatest $qc_{Pred}(b(i))$ value is chosen. (Point 3.b in the description of the algorithm).

To prevent an overloading of the prediction associative memory, states which lead to an impasse state and the impasse state itself are forgotten. Impasse states are the states for which no other states exist. A vector of the state which led to an impasse state, together with the desired state, compose the question vector. The impasse state is the answer vector. Both vectors are forgotten. During the learning of the prediction associative memory, incorect state sequences are forgotten. The sequence of states describing the plan is learned. Example of ID4 $\rightarrow$ ID1 from the geometric block world, see fig. 5.17:

These pairs are forgotten:

$$(s9 + ID1, s10), (s10 + ID1, s11), (s6 + ID1, s7), (s7 + ID1, s8), (s8 + ID1, s13),$$

$$(s13 + ID1, s14).$$

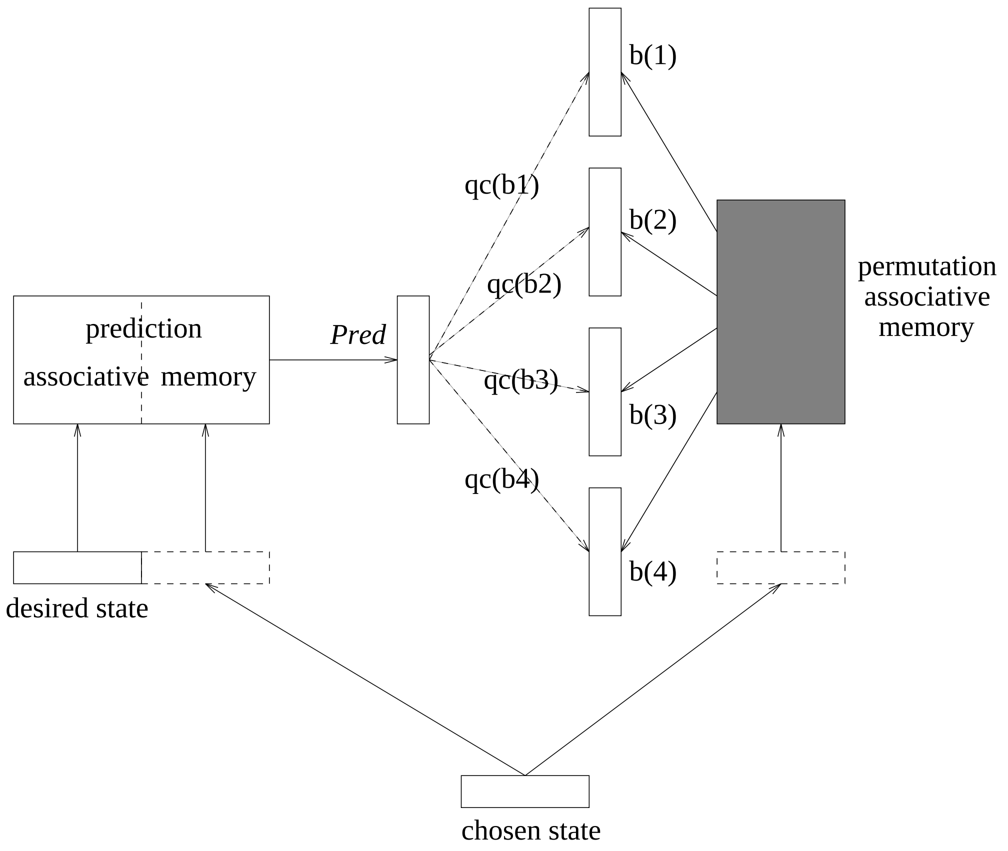


Figure 5.22: The permutation associative memory determines $b(i)$ states of a chosen state. At the same time, the prediction associative memory determines with the soft threshold strategy the answer pattern $Pred$ to this chosen state together with the desired state. The unit with the greatest $qc_{Pred}(b(i))$ value is chosen.

These pairs are learned:

$$(s0 + ID1, s1), (s1 + ID1, s2), (s2 + ID1, s3), (s3 + ID1, s4), (s4 + ID1, s5),$$

$$(s5+ID1, s18), (s18+ID1, s19), (s19+ID1, s20), (s20+ID1, s21), (s21+ID1, s22),$$

$$(s22 + ID1, s23).(s23 + ID1, ID1).$$

The behaviors of the permutation associative memory, the pattern matcher and the backtracker are changed accordingly.

Permutative Associative memory modernized updated is:

2. *No answer* If no answers exist. The question vector composed of the marked unit of layer $l-1$ plus the desired state and the answer vector of the chosen

unit of layer l are forgotten by the prediction associative memory. The chosen unit of the layer l is disabled.

The pattern matcher updated behavior is:

1. *Desired state* If a desired state is reached by one of the units of the layer l the computation succeeds. The sequence of marked units is learned by the prediction associative memory.

2. *Chose* can vary depending on the learning strategy of the prediction associative memory

   - For all $b(i)$ $i \in \{1, .., s\}$ staets represented in a layer the unit whose state has the greatest $qc^2_{desired\ state}(b(i))$ value is chosen.
   - For all $b(i)$ $i \in \{1, .., s\}$ states represented in a layer the unit whose state has the greatest $qc_{Pred}(b(i))$ value is chosen.

The backtracker updated behavior is:

1. *Backtrack* If all units of a layer l are disabled. Enable all units of a layer l. The question vector composed of the marked unit of layer $l-2$ plus the desired state and the answer vector of the marked unit of layer $l-1$ are forgotten by the prediction associative memory. Disable the marked unit of the layer $l-1$. Activate layer $l = l - 1$.

Learning strategy

Either the learning phase and the retrieval phase of the prediction associative memory are separated or they are not. If the phases are separated, a kind of "teacher" exists in the learning phase. An observer chooses relevant examples. The pattern heurisitic acts as a superviser who guides the search. This kind of learning is called supervised learning. If there is no separation, then the prediction associative memory learns by experience of failures and successes. The prediction heuristic is improved during learning. This kind of learning is called "unsupervised" learning.

Supervised learning

Nine examples of problems were chosen. The results of $h0, h1, h2$ of steps, plan length and backtracking steps are determined as shown in tables 5.4, 5.5, 5.6. After that, three learning sessions were performed. In all three learning sessions the $h2$ heuristic guided the search. In the l learning session the example ID1 $\rightarrow$ ID5 was learned. It was chosen because of its complexity. In the *all* learning session all 9 examples were learned. In the *min* learning session the four examples were chosen: ID4 $\rightarrow$ ID5, ID5 $\rightarrow$ ID1, ID3 $\rightarrow$ ID2, ID2 $\rightarrow$ ID4. The four examples were chosen because all four different desired states were included. Another reason is because, when the $h2$ heuristic was used, no backtracking occured during the search. After "learning" the steps, plan length, and backtracking steps were determined using the prediction heuristic as shown in tables 5.4, 5.5, 5.6. The *min* learning strategy led to a significant improvement in the required steps, plan length and backtracking steps compared to blind search $h0$. The *all* led to a significant improvment in required steps. The mean value of required steps using the $h2$ heuristic is lower than the mean value resulting from the *all* learning strategy, however not significantly. The p values were determined by a paired sample t test. Significant for $p < 0.05$ by convention.

task	h0	h1	h2	l	all	min
ID4 → ID5	10	34	4	16	10	4
ID1 → ID5	22	32	12	10	22	10
ID2 → ID5	14	24	24	20	8	8
ID3 → ID5	8	36	16	14	8	8
ID4 → ID1	24	12	24	6	24	6
ID5 → ID2	14	14	14	28	12	12
ID5 → ID1	12	10	10	10	10	10
ID3 → ID2	20	8	8	24	20	8
ID2 → ID4	10	8	4	4	4	4
mean	14.89	19.78	12.89	14.67	13.11	7.78
sdev	5.75	11.72	7.49	8.12	7.08	2.73
p	-	0.17	0.22	0.48	**0.034**	**0.003**

Table 5.4: Required steps.

task	h0	h1	h2	l	all	min
ID4 → ID5	10	20	4	16	10	4
ID1 → ID5	10	16	10	10	10	10
ID2 → ID5	14	22	22	20	8	8
ID3 → ID5	8	20	14	14	8	8
ID4 → ID1	12	12	12	6	12	6
ID5 → ID2	12	12	12	24	12	12
ID5 → ID1	10	10	10	10	10	10
ID3 → ID2	18	8	8	20	18	8
ID2 → ID4	10	8	4	4	4	4
mean	11.56	14.22	10.67	13.78	10.22	7.78
sdev	2.96	5.43	5.48	6.82	3.8	2.73
p	-	0.14	0.33	0.15	0.09	**0.009**

Table 5.5: Plan length.

The more the associative computer knows about problems of a certain domain, the better its behavior will be in this domain. A teacher who chooses important

task	h0	h1	h2	l	all	min
ID4 → ID5	0	7	0	0	0	0
ID1 → ID5	6	8	1	0	6	0
ID2 → ID5	0	1	1	0	0	0
ID3 → ID5	0	8	1	0	0	0
ID4 → ID1	6	0	6	0	6	0
ID5 → ID2	1	1	1	2	0	0
ID5 → ID1	1	0	0	0	0	0
ID3 → ID2	1	0	0	2	1	0
ID2 → ID4	0	0	0	0	0	0
mean	1.67	2.78	1.11	0.44	1.44	0
sdev	2.5	3.7	1.9	0.88	2.6	0
p	-	0.23	0.19	0.11	0.09	**0.04**

Table 5.6: Backtracking steps.

examples of a domain and another teacher who demonstrates how to solve those problems can help the associative computer in learning, and in efficiently solving problems in this domain. Faulty information which is sometimes indicated by the pattern heuristic can be in this way prohibited. An important condition is the appropriate choice of the material. Wrong material, too much, or too little material can worsen the problem solving behavior. A good understanding of the problem space is essential for fast and effective problem solving. In our example the pattern heuristic acted as the second teacher.

Unsupervised learning

The prediction associative memory "learned" by experience of failures and successes in the nine examples. The nine examples were solved sequentially. The prediction memory learned, simultaneously to the usage of the resulting prediction heuristic (see tables 5.7, 5.8, 5.9). The column $t0$ represents the required steps without the usage of the prediction heuristic, and corresponds to the blind search $h0$. At the beginning of learning the prediction memory is empty. First example 1: ID4 → ID5 is learned, and simultaneously the prediction heuristic is used. During the next example 2:ID1 → ID5 the knowledge which was learned from the previous example is reused. This procedure is repeated with the remaining examples. The stages are repeated sequentially another nine times. However, already at the end of the third stage, $t3$, an equilibrium for the needed steps, plan length and the backtracking steps occurs (see tables 5.7, 5.8, 5.9).

A three dimensional bar chart ilustratees the table 5.7 (see fig.5.23). The development of the unsupervised learning can be illustrated more clearly by graphs of the mean values of the required steps, plan length and backtracking steps. The scale effects are suppressed by the added error bars to the means. Each bar represents the upper and the lower bounds of a 95 percent confidence interval

task	t0	t1	t2	t3	t4-t10
1: ID4 → ID5	10	10	10	10	10
2: ID1 → ID5	22	18	22	22	22
3: ID2 → ID5	14	12	22	8	8
4: ID3 → ID5	8	14	14	8	8
5: ID4 → ID1	24	24	24	24	24
6: ID5 → ID2	14	30	12	12	12
7: ID5 → ID1	12	16	10	10	10
8: ID3 → ID2	20	30	20	18	18
9: ID2 → ID4	10	4	4	4	4
mean	14.89	17.56	15.33	12.89	12.89
sdev	5.75	8.93	6.93	6.86	6.86
p	-	0.15	0.38	**0.02**	**0.02**

Table 5.7: Required steps.

task	t0	t1	t2	t3	t4-t10
1: ID4 → ID5	10	10	10	10	10
2: ID1 → ID5	10	16	10	10	10
3: ID2 → ID5	14	12	20	8	8
4: ID3 → ID5	8	14	14	8	8
5: ID4 → ID1	12	12	12	12	12
6: ID5 → ID2	12	18	12	12	12
7: ID5 → ID1	10	16	10	10	10
8: ID3 → ID2	18	18	18	18	18
9: ID2 → ID4	10	4	4	4	4
mean	11.56	13.33	12.22	10.22	10.22
sdev	2.96	4.47	4.74	3.8	3.8

Table 5.8: Plan length.

around its means (see fig. **??**, 5.25, **??**). After the first stage there is an impairment compared to the blind search. Significant improvment to the blind search in required steps occurs after the third stage.

The size effects of forgetting and learning of the prediction memory are represented in fig. 5.27. This figure represents the number of synapses which are not zero. After the third stage the number of the synapses is reduced. The equilibrium of the the prediction associative memory is reached one stage after the stabilization in the behavior of the associative computer, at the stage four.

task	t0	t1	t2	t3	t4-t10
1: ID4 → ID5	0	0	0	0	0
2: ID1 → ID5	6	1	6	6	6
3: ID2 → ID5	0	0	1	0	0
4: ID3 → ID5	0	0	0	0	0
5: ID4 → ID1	6	6	6	6	6
6: ID5 → ID2	1	6	0	0	0
7: ID5 → ID1	1	0	0	0	0
8: ID3 → ID2	1	6	1	0	0
9: ID2 → ID4	0	0	0	0	0
mean	1.67	2.11	1.56	1.33	1.33
sdev	2.5	2.94	2.56	2.65	2.65

Table 5.9: Backtracking steps.

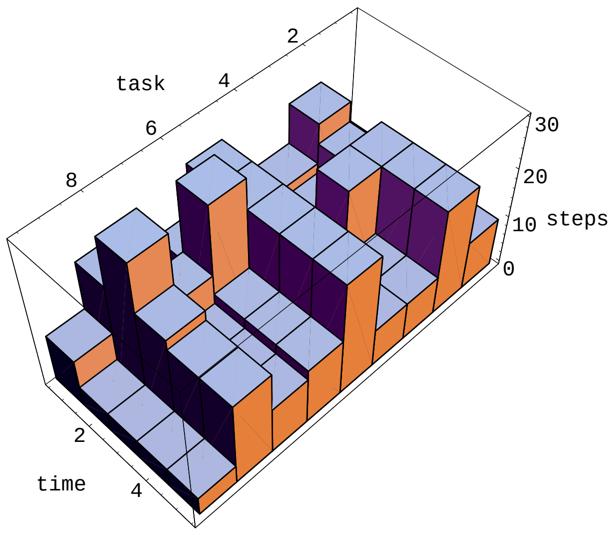


Figure 5.23: Required steps during learning.

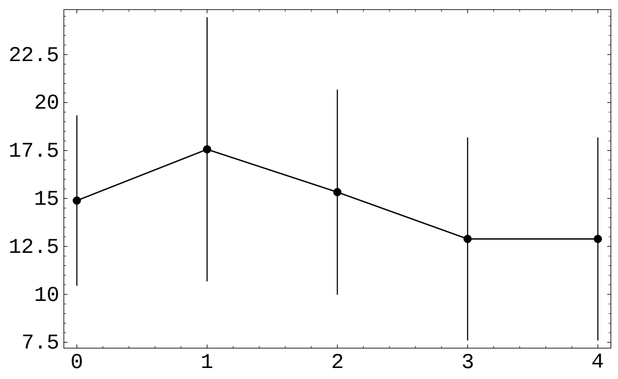


Figure 5.24: Mean steps for learning. *t3-t10: mean=12.89, sdev=6.86,* ***p=0.02***

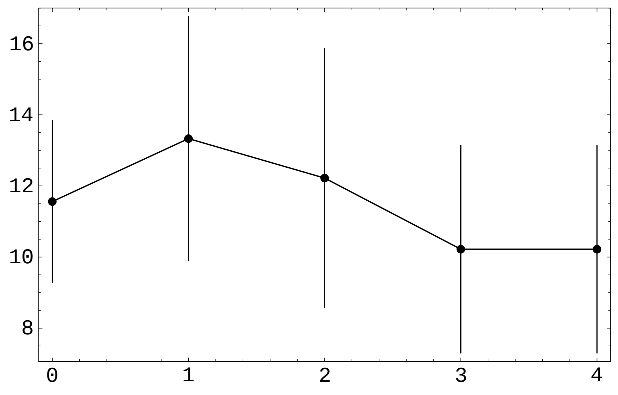


Figure 5.25: Mean plan length for learning. *t3-t10: mean=10.22, sdev=2.96*

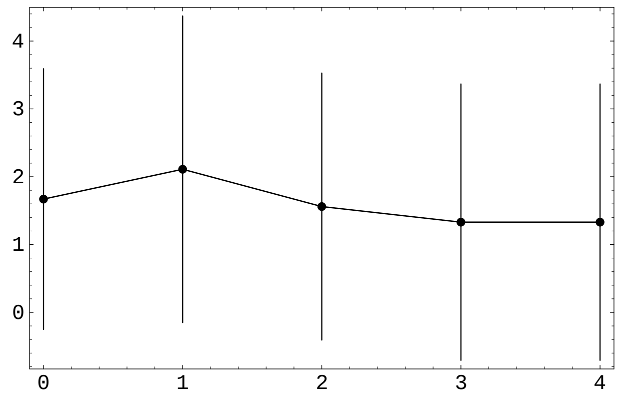


Figure 5.26: Mean backtracking steps for learning. *t3-t10: mean=1.33, sdev=2.65*

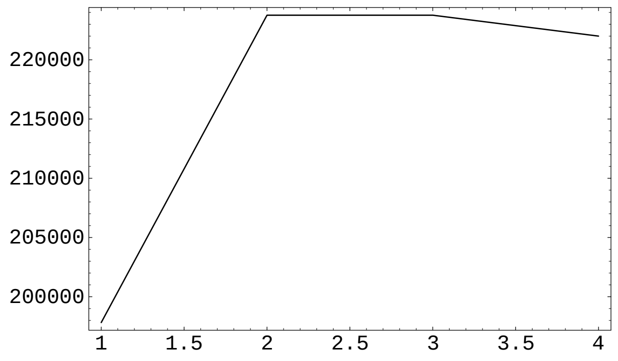


Figure 5.27: The number of synapses which are not zero during learning. *t4-t10: 222000 of* 1.99101^8 *synapses are not zero (0.1115%).*

The associative computer learns from experience if no additional information from a teacher is available. The associative computer learns to prohibit faults. This kind of learning is tedious, and different problems indicate contradictory recommendations. Despite the low load of the prediction associative weight matrix, an overlearning occurs. This is at first glance a contradiction, but by closer examination one recognizes that important information is concentrated in a small area of the prediction associative memory (see chapter 6, section about prediction associative memory). This is because during problem solving the world changes only minimally. Those changes are represented in the prediction associative memory and are concentrated in a small area of the weight matrix. Many more pattern are stored than during the supervised learning. Some learning sequences are needed until the stabilization of the behavior of the associative computer. After the first learning session, an impairment is present while an improvement is present in the succeeding steps. The improvement comes along with the gain followed by the reduction of the weights of the prediction associative memory. The behavior of the associative computer after the learning is comparable to the pattern heuristic.

Context information

A problem can have diverse solutions. The same state in different sequences can lead to different folowing states despite the same desired state. These different states are dependent on the initial state. This case can only occur when the solutions are not optimal. Nevertheless, the adherence of the initial state to the question vector could lead to the improvement of unsupervised learning. The improvement could result from the detachment of particular solutions. This kind of prediction heuristic is called the "context prediction" heuristic. The prediction associative memory learns and is used as before, however this time with the addition of the context. The question vectors are composed by concatenation of the pattern representation of the state, the initial state, and the desired state. The answer vectors correspond to the patterns of the next states (see fig. 5.28).

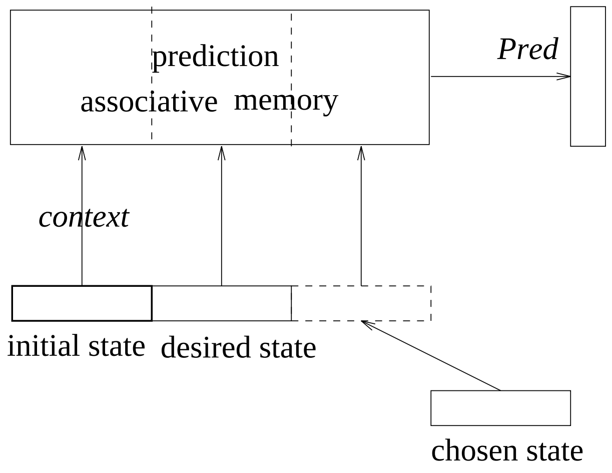


Figure 5.28: The question vectors are composed by concatenation of the pattern representation of the state.

The context unsupervised learning led, howeve, to no significant improvement in this example, and instead led to an impairment (see fig. 5.29,??, 5.31, ??, 5.33).

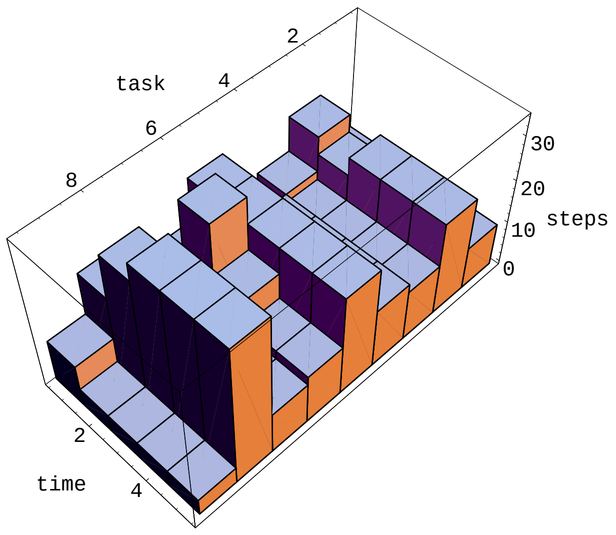


Figure 5.29: Required steps with context during learning.

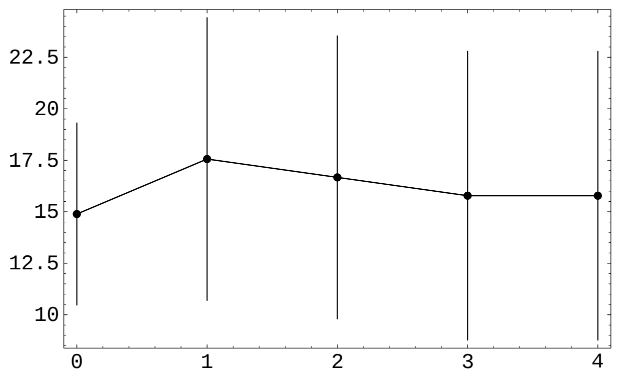


Figure 5.30: Mean steps for learning. *t3-t10: mean=15.78, sdev=9.12, p=0.33*

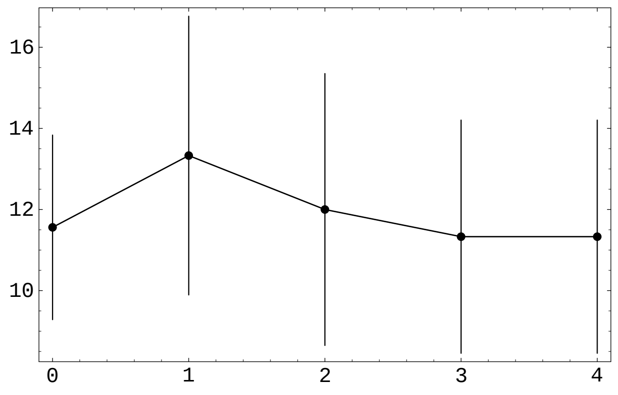


Figure 5.31: Mean plan length for learning. *t3-t10: mean=11.33, sdev=3.74*

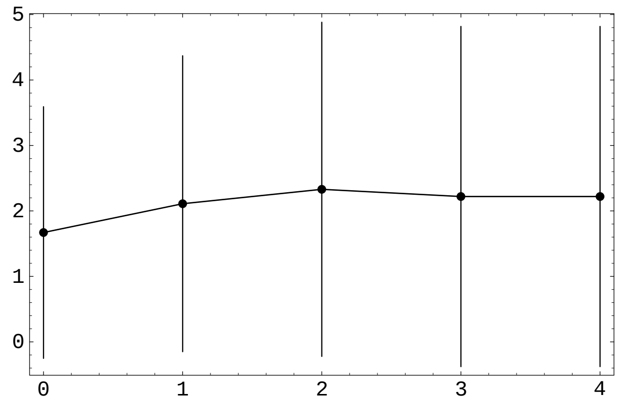


Figure 5.32: Mean backtracking steps for learning.*t3-t10: mean=2.22, sdev=3.38*

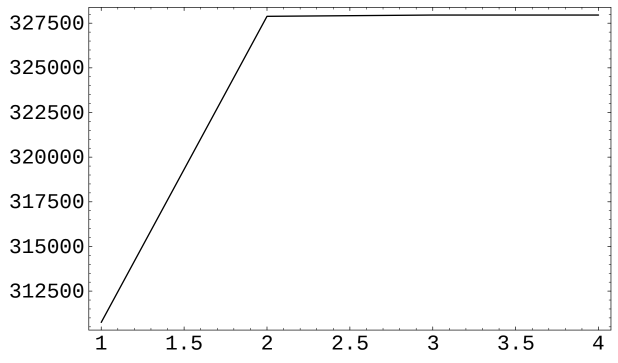


Figure 5.33: The number of synapses which are not zero during learning. *t3-t10: 327956 of* 2.98801^8 *synapses are not zero (0.1098%).*

The size effects of forgetting and learning of the prediction memory are represented in fig. 5.33. The size of the prediction associative memory with context is much larger then the size of the prediction associative memory without context, because the initial states are also stored. However, the percentage of synapses unequal to zero of the prediction associative memory with context is nearly the same as of the prediction associative memory without context.

Unsupervised learning and delete

The size effects of the forgetting and learning of the prediction memory are fractional in fig. 5.27 and absent in fig. 5.33. To determine if the delete function is more appropriate than the forgetting function to learn the prediction heuristic the foregoing experiments are repeated. The results without context are better then those with context. However, the results without context are bad. The delete function eliminates too much knowledge (see fig.**??**). The prediction heuristic can not work (see fig. **??**, 5.35, **??**, 5.37). The delete function is unable to learn and to improve the prediction heuristic. In the following examples, therefore, the delete function is never used.

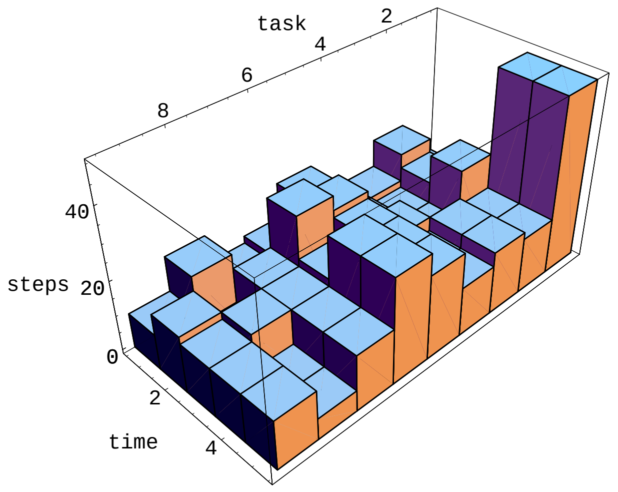


Figure 5.34: Required steps without context during delete learning.

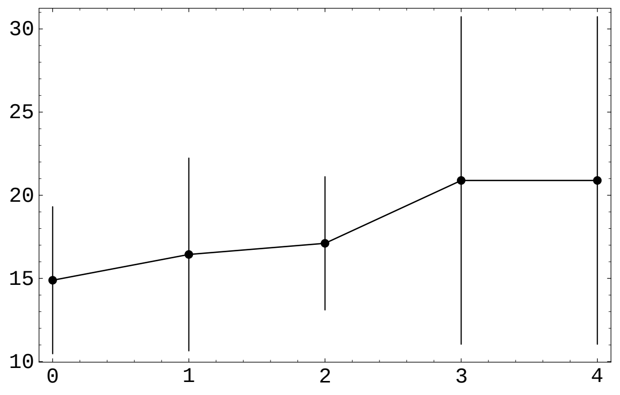


Figure 5.35: Mean steps for learning. *t3-t10: mean=20.89, sdev=12.81, p=0.14*

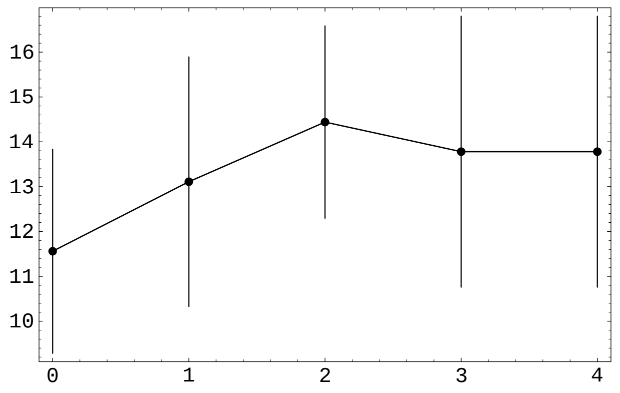


Figure 5.36: Mean plan length for learning. *t3-t10: mean=13.78, sdev=3.93*

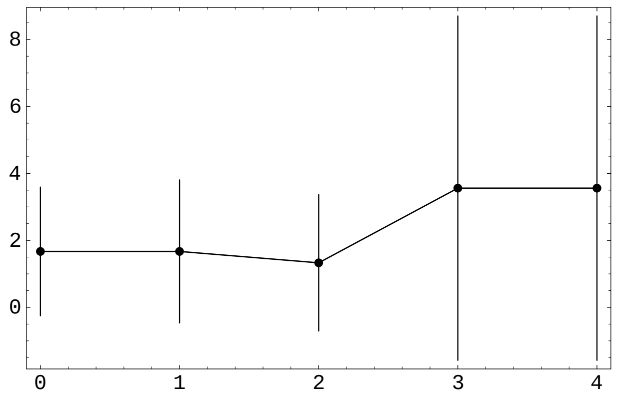


Figure 5.37: Mean backtracking steps for learning. *t3-t10: mean=3.56, sdev=6.69*

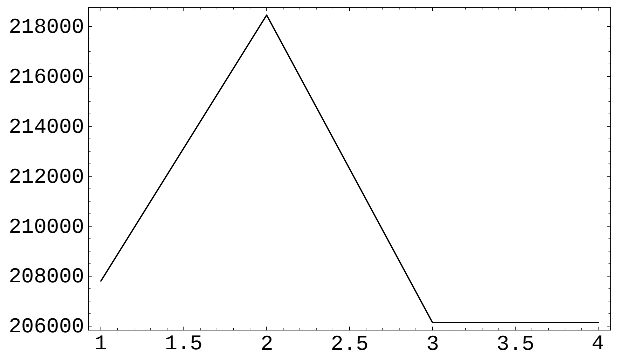


Figure 5.38: The number of synapses which are not zero during learning. *t3-t10: 206148 of* 1.99101^8 *synapses are not zero (0.1035%).*

Unsupervised learning and initialization

Only in the first example is the $h2$ heuristic used because the prediction associative memory is empty at the start of learning. ID4 $\rightarrow$ ID5 is learned, and simultaneously the $h2$ heuristic is used. During the next example (ID1 $\rightarrow$ ID5) and the subsequent examples, however, the prediction heuristic is used. The predicition heuristic continues to be used during all the subsequent stages. Two experiments are performed: without context and with context. The initialized learning without context led to no significant improvment. At the end of the second stage, an equilibruium for the needed steps, plan length, and the backtracking steps occured (see fig. 5.39, 5.40, **??**, 5.42).

The size effects of the forgetting and learning of the prediction memory are represented in fig. **??**. The equilibrium of the prediction associative memory is reached two stages after the stabilization in the behavior of the associative computer, at the stage four.

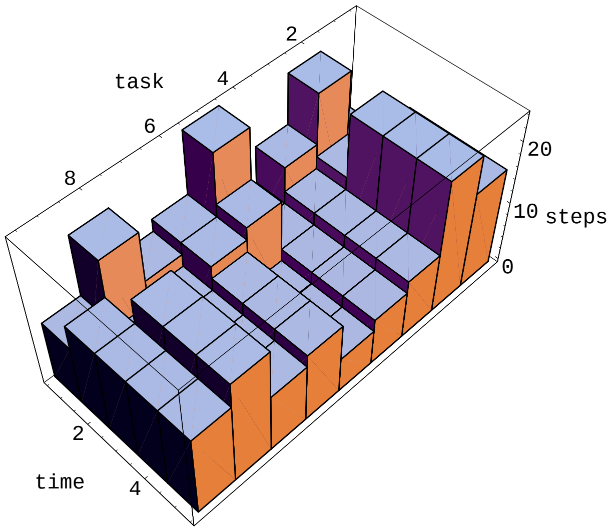


Figure 5.39: Required steps with $h2$ initialization during learning.

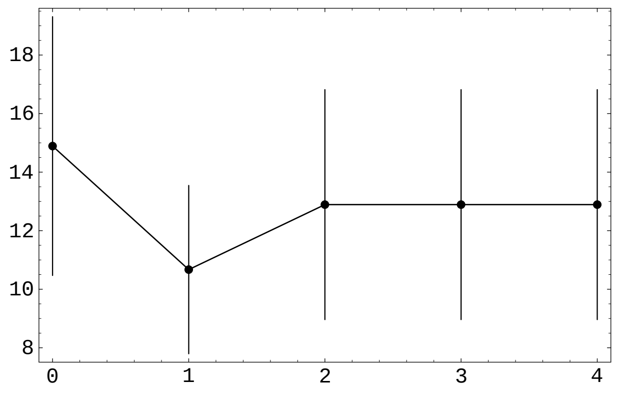


Figure 5.40: Mean steps for learning. *t2-t10: mean=12.89, sdev=5.11, p=0.2*

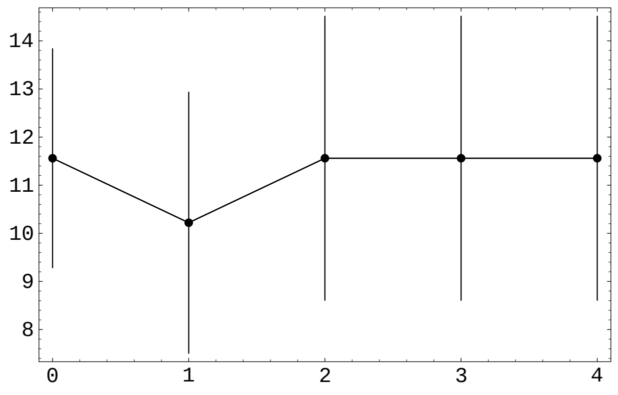


Figure 5.41: Mean plan length for learning. *t2-t10: mean=11.56, sdev=3.84*

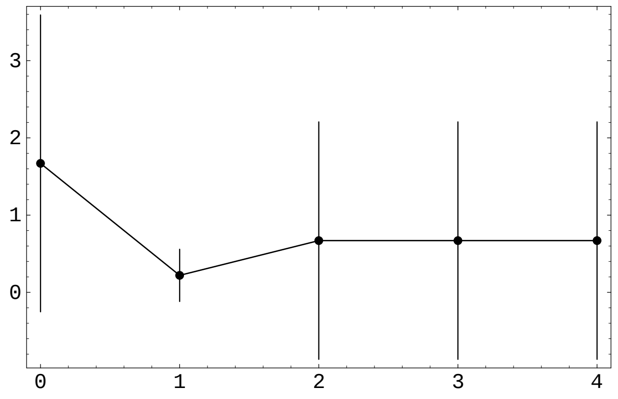


Figure 5.42: Mean backtracking steps for learning. *t2-t10: mean=0.67, sdev=2*

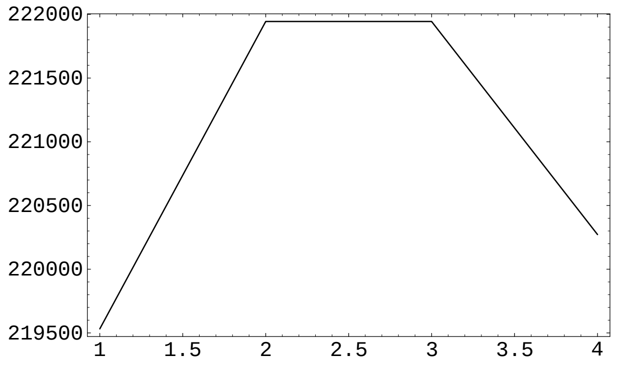


Figure 5.43: The number of synapses which are not zero during learning. *t4-t10: 220272 of* 1.99101^8 *synapses are not zero (0.1106%).*

The initialized learning with context led, however, to a significant improvment. At the end of the third stage, an equilibruium for the needed steps, plan length, and the backtracking steps occured (see fig. 5.44, **??**, 5.46, **??**).

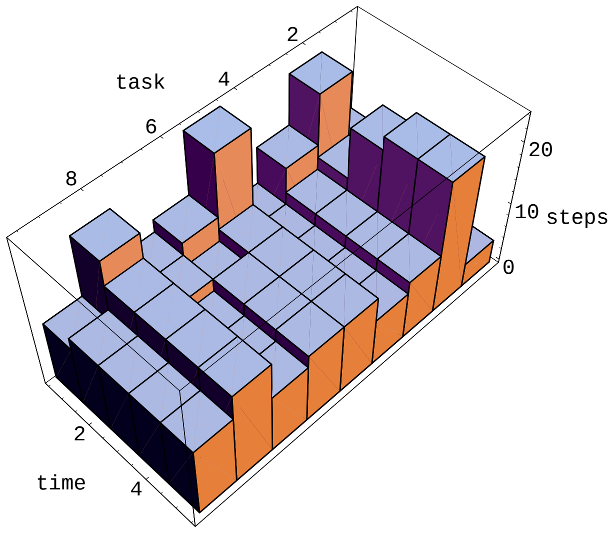


Figure 5.44: Required steps with context with $h2$ initialization during learning.

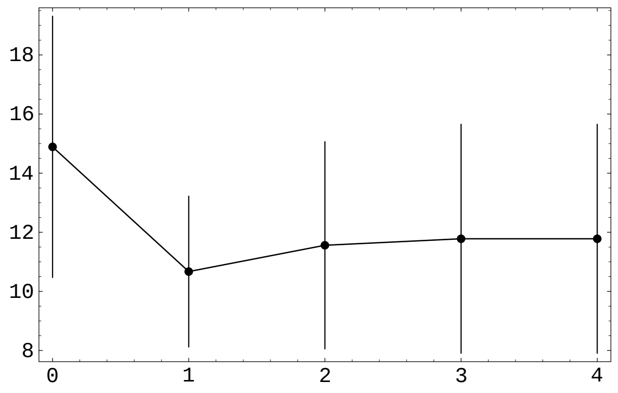


Figure 5.45: Mean steps for learning. *t3-t10: mean=11.78, sdev=5.04,* ***p=0.027***

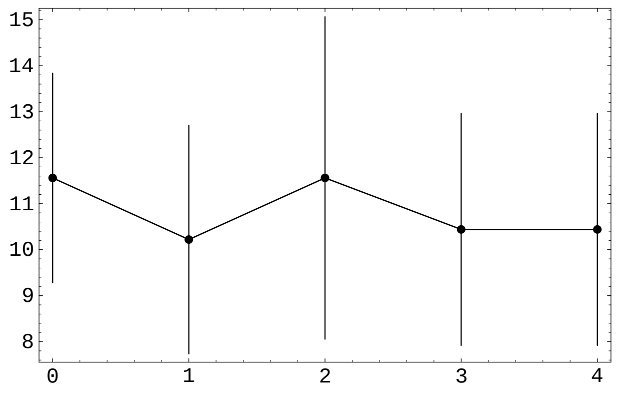


Figure 5.46: Mean plan length for learning. *t3-t10: mean=10.44, sdev=3.28*

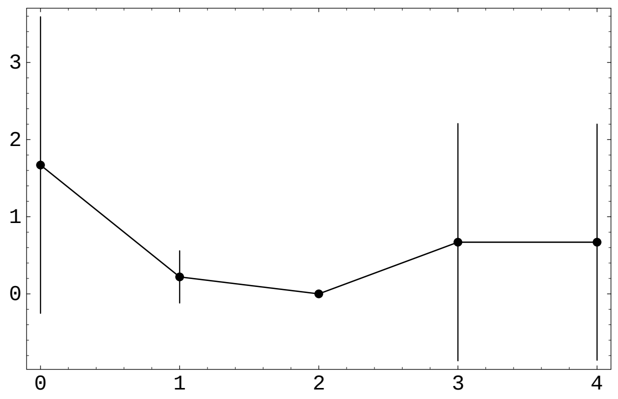


Figure 5.47: Mean backtracking steps for learning. *t3-t10: mean=0.67, sdev=2*

The size effects of the forgetting and learning of the prediction memory are represented in fig. 5.48. The equilibrium is reached one stage after the stabilization in the behavior of the associative computer, at the stage four.

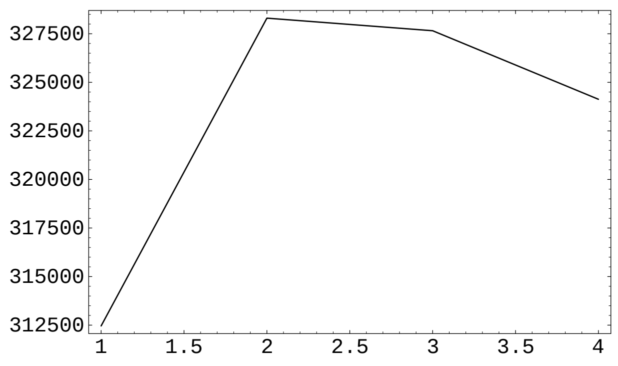


Figure 5.48: The number of synapses which are not zero during learning. *t4-t10: 324128 of* 2.98801^8 *synapses are not zero (0.1085%).*

5.3.7 Summary of the first results of the prediction heuristic

The results of the experiments for the determination of the prediction heuristic are dependent on the chosen examples and on the order of learning. Nevertheless, by these small examples, one gets the initial impression that the prediction heuristic works (see tables 5.10, **??**, 5.12). The best results were obtained with supervised learning. Four examples were chosen in *min*. They covered other examples and did not overload the prediction associative memory. Unsupervised learning (uns) was superior compared to supervised learning of all examples *all*. The context unsupervised learning and initialization (cui) led to an even better improvement. This improvement could result from the separation of particular solutions. Defacto context can help by the separation of particular solutions, but it is strongly dependent on the examples and the initialization of the prediction associative memory.

	h0	min	cui	uns	all
mean	14.89	7.78	11.78	12.89	13.11
sdev	5.75	2.73	5.04	6.86	7.08
p	-	0.003	0.027	0.02	0.034

Table 5.10: Required steps, the best learning results.

	h0	min	cui	uns	all
mean	11.56	7.78	10.44	10.22	10.22
sdev	5.75	2.73	3.28	3.8	3.8

Table 5.11: Plan length, the best learning results.

	h0	min	cui	uns	all
mean	1.67	0	0.67	1.33	1.44
sdev	2.5	0	2	2.65	2.6

Table 5.12: Backtracking steps, the best learning results.

The nine chosen examples (see table 5.4) did not cover all 30 possible combinations of the initial and desired states. The prediction associative memory which was learned by those nine examples did not bring any improvment to the 30 possible combinations (see table **??**).

	h0	l^{small}	all^{small}	min^{small}
mean	14.4	16.47	13.6	14.4
sdev	10.2	10.47	8.72	9.99
p	-	0.18	0.23	0.34

Table 5.13: Required steps for all 30 combinations after supervised learning by nine examples.

The linear combination of the pattern heuristic and the prediction heuristic led to an impairment compared to the blind search strategy. Suggestions of both heuristics were often inconsistent.

The more knowledge about problem solving is present, the better the behavior of the associative computer. A good education resulting from supervised learning with a competent teacher give the best results. Sequential learning during unsupervised learning leads to better results than supervised learning of all examples guided by the pattern heuristic. In a closed domain in which learning take place, the results are comparable to the pattern heuristic. The hints of the real world in the form of pictorial representation require, however, fewer resources than learning from experience. The usage of an associative computer which learned from experience as a teacher to optimize the problem behavior of another associative computer was shown in part in the experiments with unsupervised learning and initialization.

5.3.8 Noise

The handling of noise by the associative computer will now be examined. The initial states are disturbed to examine the behavior during chaining. First, the valid associations are recognized by the permutation associative memory, $threshold_{part} = 0.7$ and $threshold_{whole} = 0.87$ (see paragraph 5.3.1). If the threshold values are too low, wrong associations are executed. Secondly, during the computation the states are changed by stepwise elemination of noise. In the experiments pixels of the associative field describing an object are either deleted (see fig. 5.49, d-I2) or added (see fig. 5.49, a-I2). The bars describing the position are shifted (see fig. 5.49, s-I2) or the three kinds of noise are combined (see fig. 5.49, c-I2, c-I3, c-I4). In the table 5.14 the required steps depending on the kind of noise are shown. The *min* is the prediction heuristic resulting from the supervised learning (see table 5.4). The collapse of the associative computer is indicated by *none*. The collapse results either from not recognizing any assocaitions or by producing mirage states. The prediction heuristic helps to eleminate the production of mirage states.

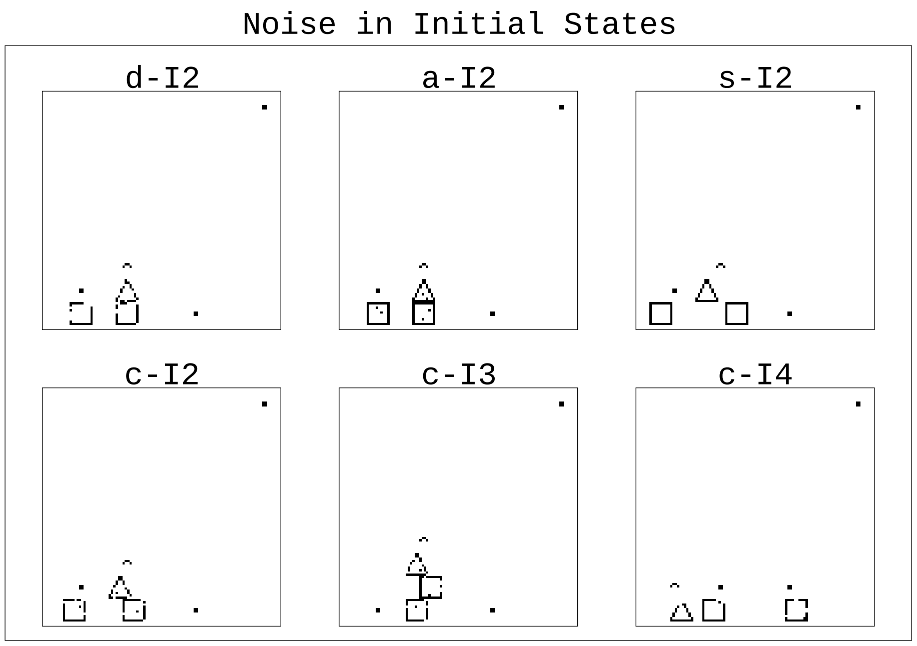


Figure 5.49: The initial states for the task geometric blocks with noise.

type of noise	h0	min
5 pixel deleted in a field	18	8
10 pixel deleted in a field	none	none
d-I2, 10 pixel for a square, 7 for a triangle deleted	18	8
10 pixel for a square, 8 for a triangle deleted	18	**none**
a-I2, 2 pixel added	48	8
3 pixel added	none	none
1 pixel added, 5 pixel deleted	**none**	8
s-I2, moved 7 pixel in the horizontal	**none**	8
moved 8 pixel in the horizontal	none	none
moved 1 pixel in the vertical	none	none
c-I2, 5 pixel deleted, 1 pixel added, moved 3 pixels in the horizontal	48	8

Table 5.14: Different noise types for the task ID2 → ID5, required steps.

task	h0	l	min
c-I2 → ID5	48 *(+34)*	8 *(-12)*	8 *(0)*
c-I2 → ID4	14 *(+4)*	6 *(+2)*	16 *(+12)*
c-I3 → ID5	136 *(+128)*	8 *(-6)*	12 *(+4)*
c-I3 → ID2	122 *(+102)*	100 *(+76)*	120 *(+112)*
c-I4 → ID1	94 *(+70)*	24 *(+18)*	14 *(+8)*
mean	82.8 *(+67.6)*	29.2 *(+15.6)*	34 *(+27.2)*
sdev	51.08 *(6.72)*	40.24 *(8.65)*	48.17 *(1.79)*

Table 5.15: Steps.

task	h0	l	min
c-I2 → ID5	18 *(+4)*	8 *(-12)*	8 *(0)*
c-I2 → ID4	14 *(+4)*	6 *(+2)*	16 *(+12)*
c-I3 → ID5	28 *(+20)*	8 *(-6)*	12 *(+4)*
c-I3 → ID2	20 *(+2)*	24 *(+4)*	22 *(+14)*
c-I4 → ID1	18 *(+6)*	24 *(+18)*	14 *(+8)*
mean	19.6 *(+7.2)*	14 *(+1.2)*	14.4 *(+7.6)*
sdev	5.18 *(3.85)*	9.17 *(7.56)*	5.18 *(1.79)*

Table 5.16: Plan length.

task	h0	l	min
c-I2 → ID5	15 *(+15)*	0 *(0)*	0 *(0)*
c-I2 → ID4	0 *(0)*	0 *(0)*	0 *(0)*
c-I3 → ID5	59 *(+59)*	0 *(0)*	0 *(0)*
c-I3 → ID2	51 *(+50)*	38 *(+36)*	49 *(+49)*
c-I4 → ID1	38 *(+32)*	0 *(0)*	0 *(0)*
mean	32.6 *(+31.2)*	7.6 *(+7.2)*	9.6 *(+9.6)*
sdev	24.6 *(2.61)*	16.99 *(0.89)*	21.47 *(0)*

Table 5.17: Backtracking steps.

Because the states are changed by stepwise elimination of noise, the associative computation takes more steps. This is because a picture corresponding to a state with noise and without noise representing the same scene is not the same (see fig. **??**). In table **??** the results of $h0$, l, min of steps, plan length, and backtracking steps are determined as shown in the table. l and min were learned before (see table 5.4). The prediction heuristic helps despite the noise which disturbs the prediction associative memory. The value in round brackets indicates impairment via no noise (table 5.4, 5.5, 5.6).

The information supplied by sensors during problem solving can often be "noisy" or incomplete. Noise can also result from faulty hardware. For example, the cameras of a robot can supply noisy pictures, such as when the sensors of a space probe have a malfunction. In this case, it is important that the computing system does not collapse. It was shown that the associative computer is a robust model and can tolerate noise to some extent. It is an ideal model for performing problem solving computation in a robot. On the other hand every biologically plausible model which simulates human problem solving behavior performed by the brain should tolerate noise to some extent, as noise is present in the real human brain.

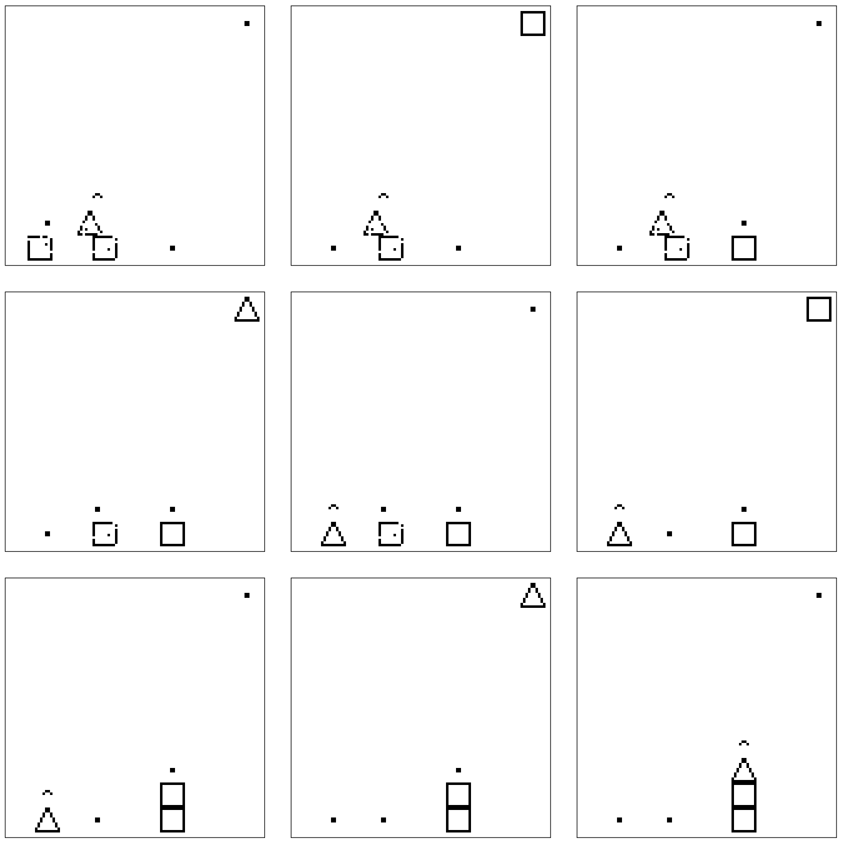


Figure 5.50: Planning of the task cI2 → ID3 with prediction heuristic *min*.

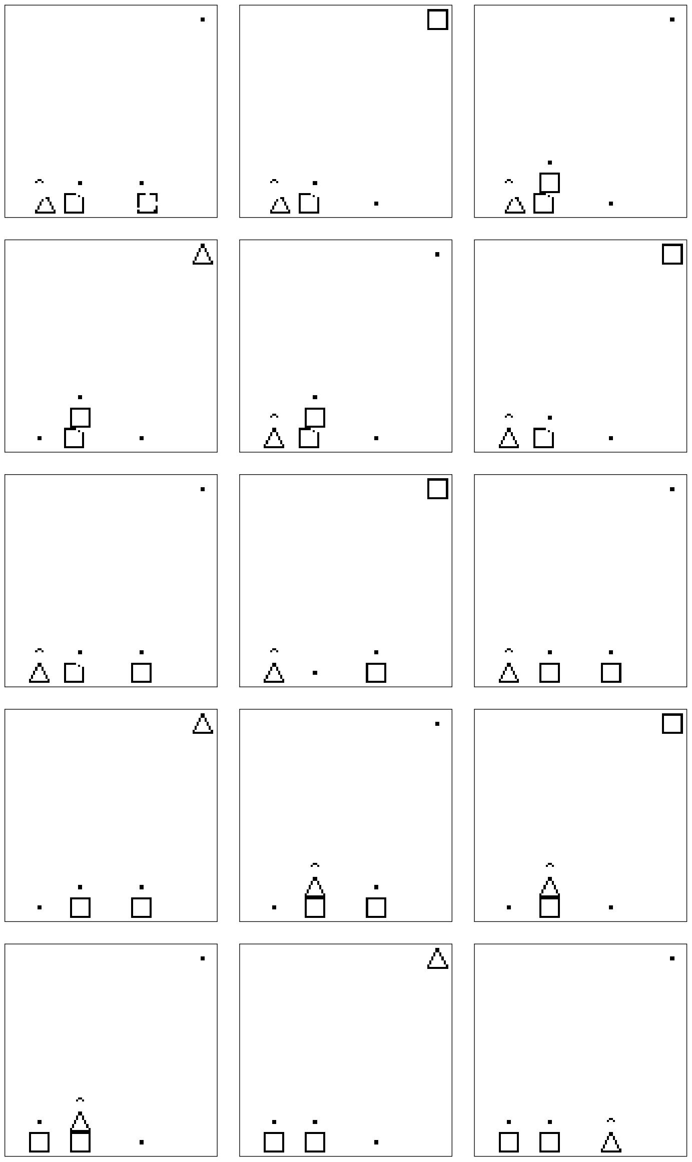


Figure 5.51: Planning of the task cI4 → ID1 with prediction heuristic *min*.

Not seen objects

"Not seen" objects corespond to objects which were not learned, and objects at positions which were also not learned by the permutation associative memory (see fig. 5.52). These additional objects are not recognized, and treated as if not present by the permutation associative memory. The pattern heuristic is not influenced by them (see table **??**, ID1 → ID2, ID2 → ID1). However, they represent noise for the prediction memory, namely wrong components in the question vector. The addition of known objects at known positions sometimes helps the $h2$ pattern heuristic speed up the search (see 5.53). However, it also disturbs the prediction heuristic (see table **??**, ID3 → ID4).

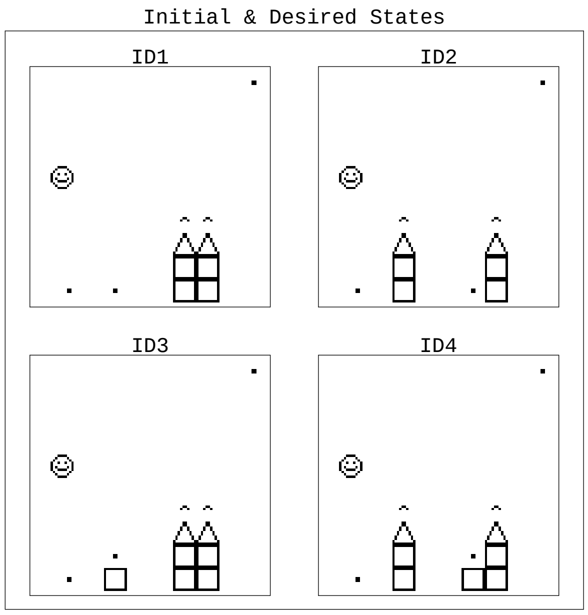


Figure 5.52: The initial and desired states with not learned objects.

task	h0	h2	l
ID1 → ID2 *(ID3 → ID5)*	8 *(0)*	16 *(0)*	8 *(-6)*
ID2 → ID1 *(ID5 → ID3)*	14 *(+6)*	16 (0)	8 *(0)*
ID3 → ID4 *(ID5 → ID3)*	6 *(-2)*	14 *(-2)*	20 *(+12)*

Table 5.18: Required steps.

task	h0	h2	l
ID1 → ID2 *(ID3 → ID5)*	8 *(0)*	14 *(0)*	8 *(-6)*
ID2 → ID1 *(ID5 → ID3)*	14 *(-6)*	14 *(0)*	8 *(0)*
ID3 → ID4 *(ID5 → ID3)*	6 *(-2)*	14 *(0)*	16 *(+8)*

Table 5.19: Plan length.

task	h0	h2	l
ID1 → ID2 *(ID3 → ID5)*	0 *(0)*	1 *(0)*	0 *(0)*
ID2 → ID1 *(ID5 → ID3)*	0 *(0)*	1 *(0)*	0 *(0)*
ID3 → ID4 *(ID5 → ID3)*	0 *(0)*	0 *(-1)*	2 *(+2)*

Table 5.20: Backtracking steps.

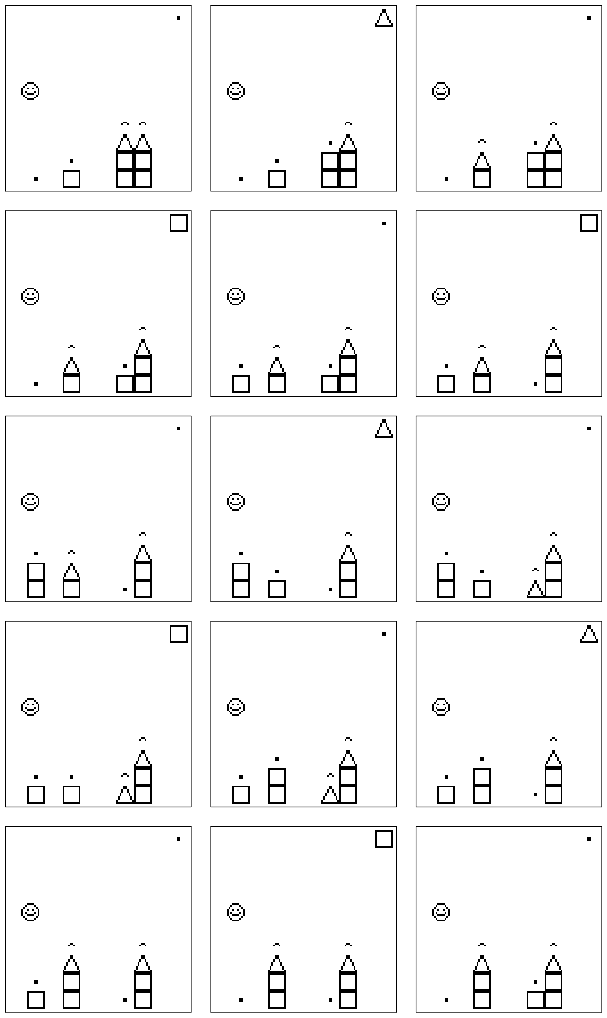


Figure 5.53: Planning of the task ID3 → ID4 with $h2$ heuristic.

A robot which was trained to solve problems with certain objects should still solve those problems even when a new, not-learned object appears. The associative computer can deal with this problem. When the objects are unimportant for the problems being solved, they are ignored. In other cases they are treated as noise.

5.3.9 Additional associations

Fifty additional associations are learned by the permutation associative memory in order to consider five additional positions on the table. The blocks can now be posed on eight positions at the table and picked up and set down by a robot arm. Stacking, however, is restricted to two stacked cubes with a pyramid on top. It is now posssible to examine the behavior of an associative computer when it has to deal with more objects. Thirten objects are present: eight cubes and three pyramids. The state of the world is again described by a hundred times hundred pixel pictures (see fig. **??**). These pictures are represented by 20 cognitive entities.

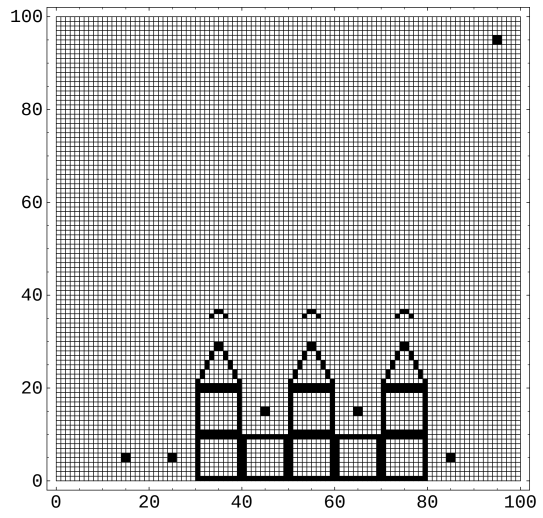


Figure 5.54: The world for the task tower.

After learning, a weight matrix (see fig. 5.55) emerges. It has the same structure as the previous matrix, but is more dense (see fig. 5.8). The additional learned positions correspond to the positions from 201 to 300 and 501 to 600 in the vectors describing the associations. Those sections in the weight matrix are more dense. The weights are repeated at those positions in the weight matrix either in the horizonthal or vertical. The four clusters in the diagonal correspond to correlations between those positions. The matrix is fully loaded. It is composed of twelve elementary blocks. The blocks still represent nearly linear correlation with peaks at the same postitions: Blocks with sum values 122 and 224 (see fig. **??**).

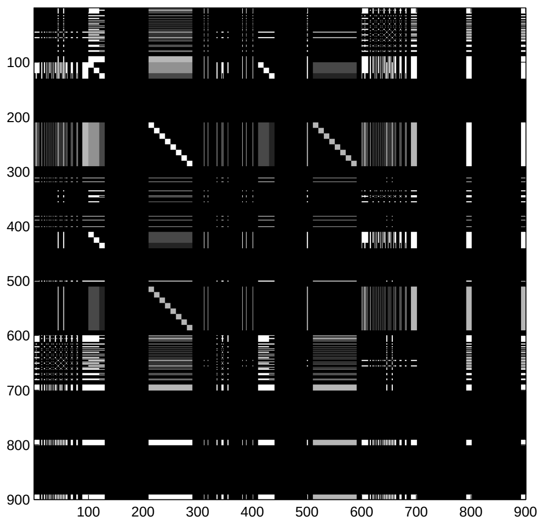


Figure 5.55: The weight matrix of the permutation associative memory for the task "tower". 77548 synapses are not zero, 9.5738% of the weight matrix.

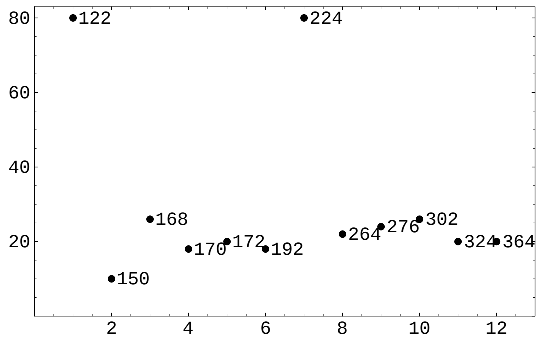


Figure 5.56: The sum of the synapses of a column. The x axis indicates the frequency of different sum values of columns. The written number corresponds to sum values.

Fig. 5.57 represents examples of initial and desired states for the geometric block world with 13 blocks as pictures.

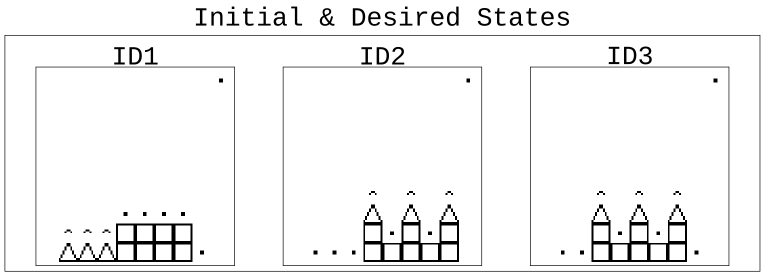


Figure 5.57: The initial and desired states for the task to build a tower.

Two examples were chosen, ID1 → ID2 and ID2 → ID3. The results of $h0, h1, h2$ of steps are determined as shown in table **??**. No backtracking occured. Because of restricted computer resources, the quantity of steps was limited to 100. The blind search could not find a solution under this restriction. The two examples were learned by the prediction associative memory, and $h2$ guided the search. The learned sequence could be recalled without errors.

task	h0	h1	h2	all
ID1 → ID2	none	62	10	10
ID2 → ID3	none	42	22	22

Table 5.21: Learning.

The prediction associative memory could generalize on two other not learned examples. A solution was found (see table 5.22). In example ID1 → ID3 the interference with the learned state ID2 of the task ID1 → ID2 is seen. A turret is built on the left corner (fig. 5.59 and **??**) instead of one position shifted from the corner. The correction of this repeated error is accomplished towards the end of the plan (see fig. 5.61).

task	h0	h1	h2	*all*
ID3 → ID2	none	58	28	*78*
ID1 → ID3	none	80	34	*36*

Table 5.22: Test.

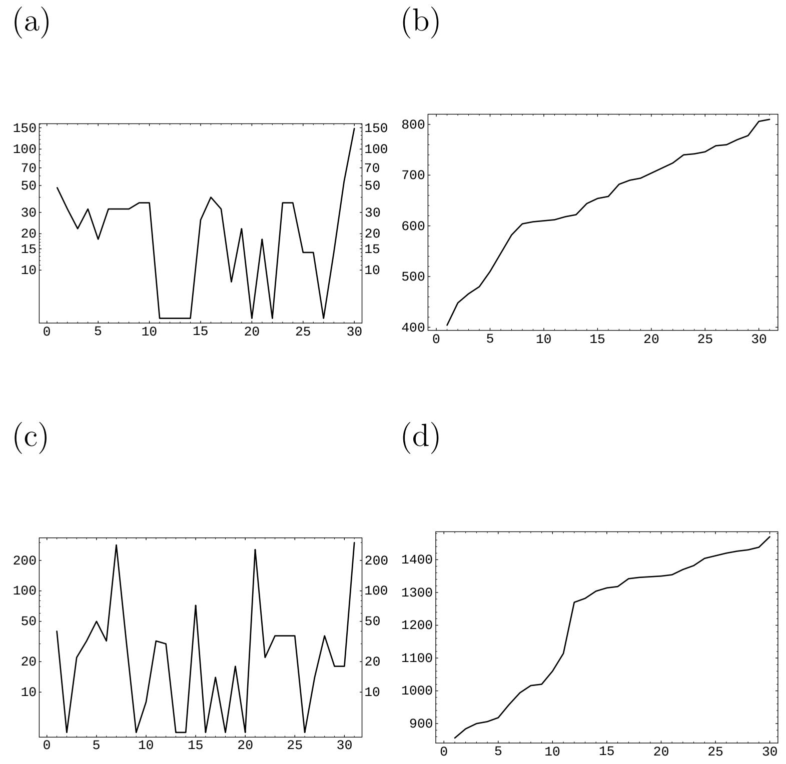


Figure 5.58: 0.4962% synapses are not zero of associative memory of the task *all*. (a) The frequency of different sum values of columns *(logarithmic plot)*. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows *(logarithmic plot)*. (d) The corresponding sum values of rows.

The prediction asymetric associative memory is dense. The structure is composed of thirty blocks in the horizontal and vertical. The distribution is ordered with strong runaway values representing peaks of correlation (see fig.**??**). The sum values are, however, nearly linear, comparable with plots b) an d) of fig. 2.4[5].

[5]Because of the high dimension of the prediction associative memory and the restricted computer resources, weight matrix diagram could not be ploted, for dimensions see table 6.29.

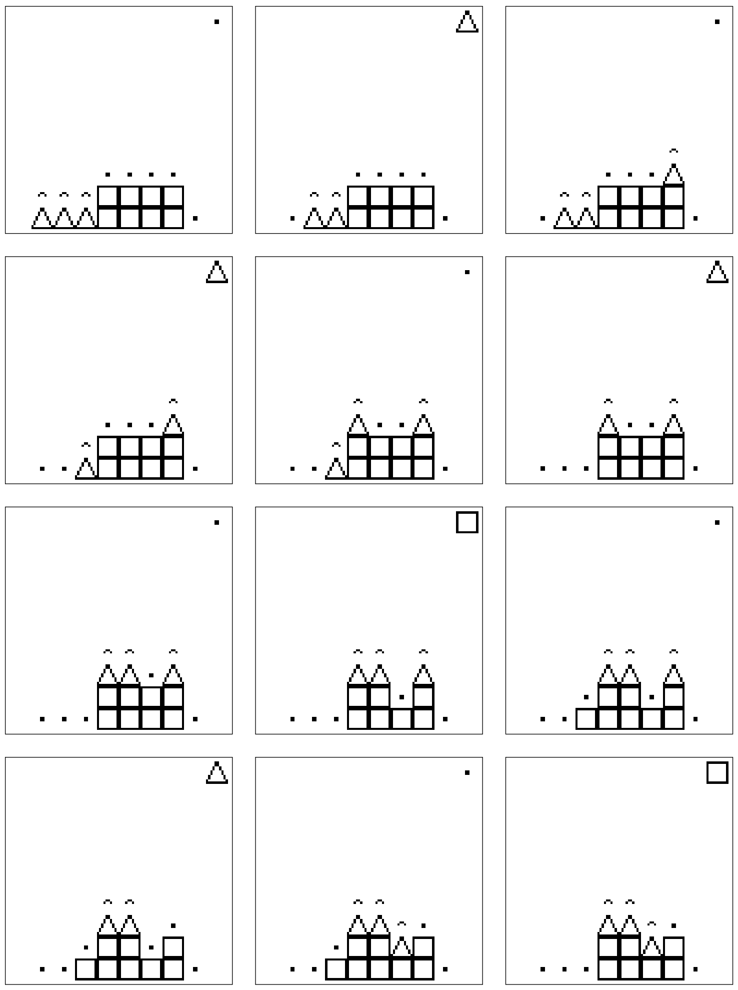


Figure 5.59: First part of the planning of the task ID1 → ID3 using the prediction *all*.

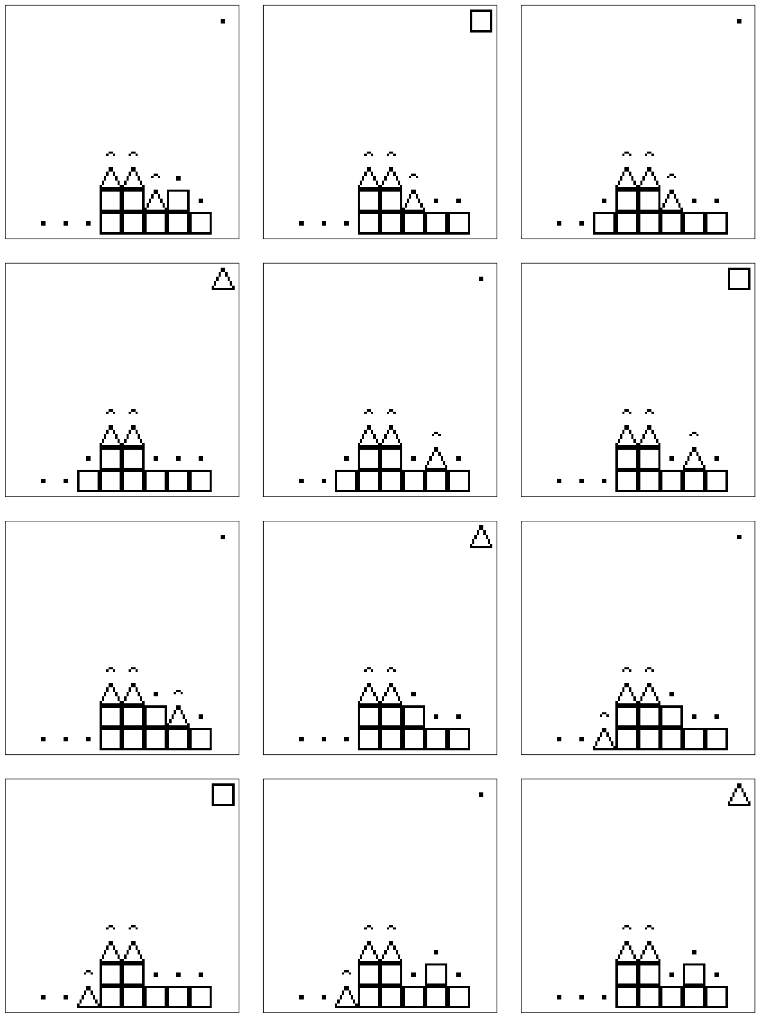


Figure 5.60: Second part of the planning of the task ID2 → ID3 using the prediction *all*.

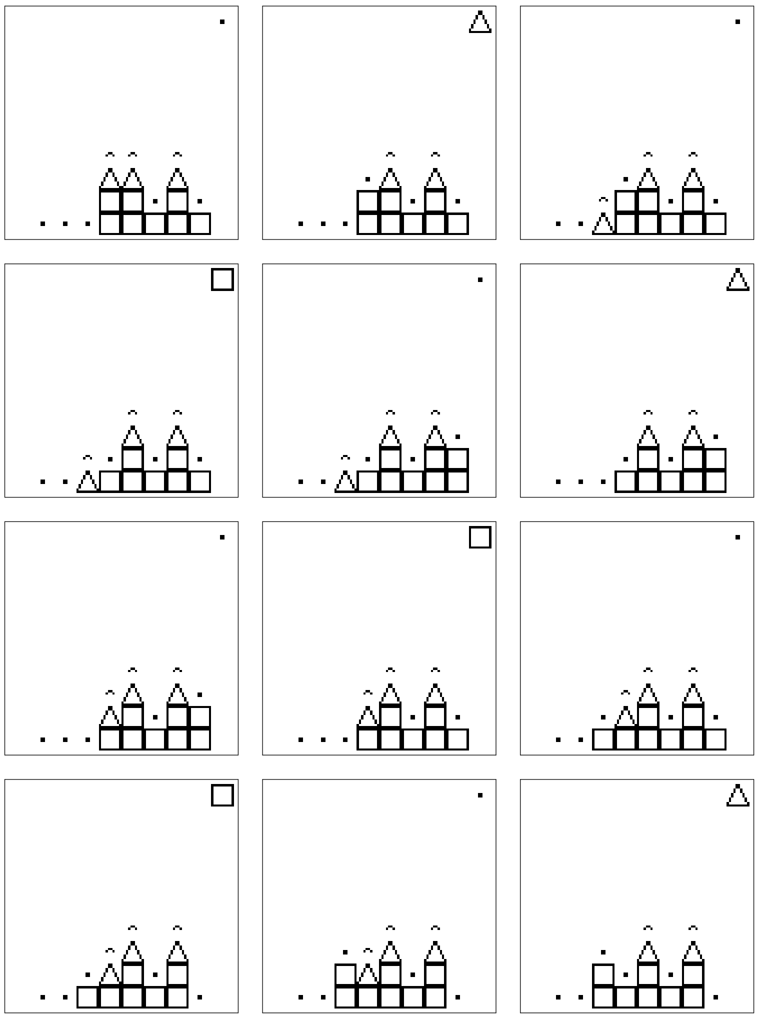


Figure 5.61: Third part of the planning of the task ID2 → ID3 using the prediction *all*.

The problem domain of the associative computer can be extended if required. This is done by the additional learning of associations by the permutation associative memory. In the tower example, the prediction associative memory learned to generalize, despite large amounts of information which differed only slightly. The required information was extracted by the prediction associative memory.

5.4 Conclusion

The associative computer, a neural model for a reaction system based on the assembly theory, was introduced. It was shown that planning can be realized by a neural architecture that does not use symbolic representation. A crucial point is the description of states by pictures. These pictures can be either represented by cognitive entities or patterns. Cognitive enty representation enables the recognition and execution of associations by the formation of permutations. These associations can be coded and represented by diverse neural network architectures. However, one of these architectures, the associative memory, offers certain advantages. Besides its high storage capacity, it is also the building block of the permutation associative memory. Permutation associative memory, motivated primarily by the RETE algorithm but also by biology and psychology, reduces both time and memory requirements. The pattern representation of pictures enables the usage of pattern heuristics. It is also the presupposition for the prediction heuristic which learns by examples, just as the availability heuristic does.

The system, the associative computer, has the same behavior as a pure symbolical reaction system (production system). It was shown how mapping of a symbolical reaction system to a neural network can be performed. For a cognitive-science scientist, it is important to understand how such computations can be performed by the human brain. Another question is whether neural systems increase our understanding of cognition or not. Are there advantages of performing this computation by neural networks or not?

An important research field in Artificial Intelligence is the usage of heuristics to speed up the search. It is not easy to define heuristic functions, as there is no rule which says how to do this. A heuristic function results automaticaly in the associative computer from the representation of knowledge by pictures. Another approach to the view of heuristic functions is shown: heuristic functions which result from the manner of description of knowledge. The first intuitive idea, that a state represented as a picture is nearer to a desired state represented as a picture, the more similar those pictures are, was statistically confirmed. In fact, pattern heuristic speeds up the search. The real world give us insights on how to solve the problem. Such an insight is used by the pictorial representation and the resulting similarity criterion. This information is not always correct, but it is generally better to obey it than to ignore it.

Another property of the associative computer is the possibility of learning with a teacher. A teacher who chooses important examples of a domain and another teacher who demonstrates how to solve those problems can help the associative computer in learning and in efficiently solving problems in this domain. Faulty information which is sometimes indicated by the pattern heuristic can be in this way prohibited. An important condition is the appropriate choice of the material.

Wrong material, too much or, too little material can worsen the problem solving behavior. A good education about the problem space is essential for fast and efficient problem solving. In our example, the pattern heuristic acted as the second teacher.

The associative computer learns from experience if no additional information from a teacher is available. It learns to prohibit faults. This kind of learning is tedious, and different problems indicate contradictory recommendations. Despite the low load of the prediction associative weight matrix, an overlearning occurs. This is, at first glance a contradiction, but by closer examination one recognizes that during problem solving, the world changes only minimally. Those changes are represented in the prediction associative memory and are concentrated in a small area of the weight matrix.

Some learning sequences are needed until the stabilization of the behavior of the associative computer. After the first learning session, an impairment is present while an improvement is present in the succeeding steps. The behavior of the associative computer after learning is comparable to the pattern heuristic. However, a good education resulting from supervised learning with a competent teacher gives the best results.

The information supplied by sensors during the problem solving can be often "noisy" or incomplete. Noise can also result from faulty hardware. For example, the cameras of a robot can supply noisy pictures, such as when the sensors of a space probe have a malfunction. In this case, it is important that the computing system does not collapse. It was shown that the associative computer is a robust model and can tolerate noise to some extent. It is an ideal model for performing problem solving computation in a robot. On the hand side every biologically plausible model which simulates human problem solving behavior performed by the brain should tolerate noise to some extent as noise is present in the real human brain.

A robot which was trained to solve problems with certain objects should still solve those problems even when a new, not-learned object appears. The associative computer can deal with this problem. When the objects are unimportant for the problems being solved, they are ignored. In other cases they are treated as noise.

The problem domain of the associative computer can be extended if required. This is done by the additional learning of associations by the permutation associative memory.

Chapter 6

Experiments

Once a problem is described using an appropriate representation, it is almost solved. The associative computer utilizes cognitive entity and pattern representation to describe problems. By using this type of internal representation the pattern and prediction heuristic is nearly defined. To show that the type of representation used is not constrained to the geometric blocks world, three other domains are analyzed. These domains were chosen because they are well known and extensively studied in the Artificial Intelligence community. In the "colored" or "marked" blocks world, the blocks differ by color, but not by form [229, 43, 196, 111]. It is shown how color can be efficiently coded. The "Sussman anomaly" is presented, and the process of building a tower of blocks is examined. In the "8-Puzzle" domain, the internal representation of numbers and the resulting pattern heuristic is examined. A final domain studied is that of a robot in a maze.

6.1 ABC blocks world

6.1.1 Permutation associative memory

In this example, blocks can be placed in three different positions and picked up and set down by a robot arm. There are three different types of blocks. They differ by attributes such as color (red, green, blue) or marks, but not by form. In AI they are traditionally called A, B, C blocks (see fig. 3.2) [111]. In our representation the A, B, C marks correspond to the marks at the corner of the counter representing the blocks. In (fig. 6.1) block C is on the table, block B on block C, and block A on block B. Block A is clear. The robot arm is represented in the upper right corner. Sixty different states are possible and fifty-four associations are learned.

The pictures are represented by seven cognitive entities. After learning, a weight matrix of dimension 900^2 emerges (see fig. 6.2). The matrix differs slightly

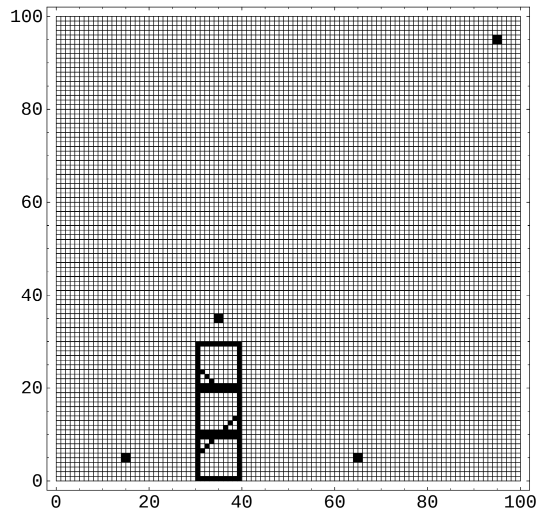


Figure 6.1: The world for the task ABC blocks.

from the matrix of the geometric blocks world. (see.fig). The differing positions correspond to the coding of the blocks: Position 1 to 100, 301 to 400 and 601 to 700.

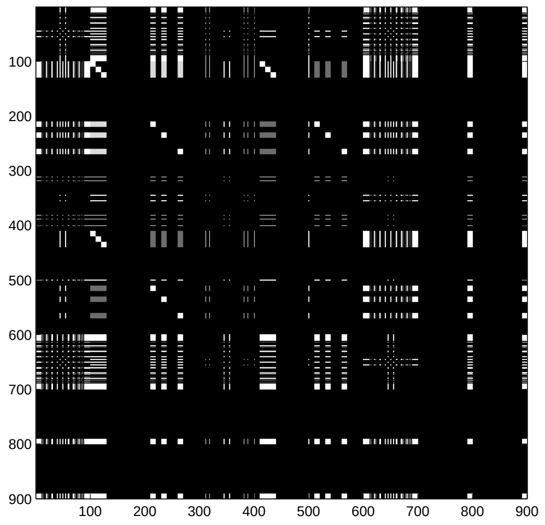


Figure 6.2: The weight matrix of the permutation associative memory for the task ABC blocks. 42774 synapses are not zero, 5.2807% of the weight matrix.

The weight matrix is composed of only seven elementary blocks (see fig 6.3). This reduction results from the fact that the different blocks are represented by vectors with the same number of ones.

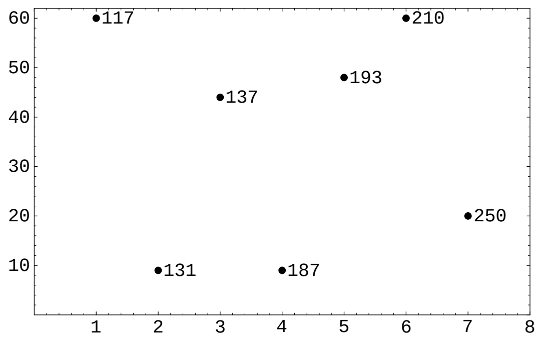


Figure 6.3: The sum of the synapses of a column. The x axis indicates the frequency of different sum values of columns. The written number correspondsto the sum values.

6.1.2 Pattern heuristics

The pattern heuristic is slightly modified in the “ABC blocks” world to improve its quality. The counter of the blocks are hidden (see fig. 6.4).

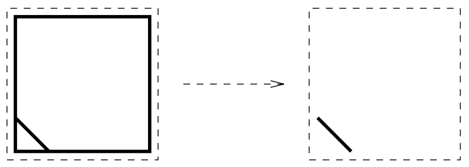


Figure 6.4: The counter of the block is hidden.

Only the reight marked blocks are considered by the heuristic function. Fig. 6.5 shows the initial and fig. 6.6 the desired state. If each of the seven states is either the initial or the desired state, then 42 combinations are possible. With these combinations both pattern heuristic functions, *h1*, and *h2*, bring a significant improvement in comparison to the blind search (see tab. 6.1, 6.2, 6.3). The part blind *h1* heuristic function brings a significant improvement in the “ABC block” world. This is because the search space is much bigger than the one in the “geometric blocks” world.

	h0	h1	h2
mean	56.29	27.24	23.38
sdev	90.79	18.26	15.6
p	-	**0.022**	**0.011**

Table 6.1: Required steps for all 42 I1-I6 and D1 combinations.

	h0	h1	h2
mean	28.91	21.43	19.57
sdev	16.62	12.22	11.84
p	-	**0.003**	**0.001**

Table 6.2: Required plan length for all 42 I1-I6 and D1 combinations.

	h0	h1	h2
mean	13.79	2.9	1.9
sdev	43.46	3.66	2.52
p	-	0.055	**0.04**

Table 6.3: Required backtracking steps for all 42 I1-I6 and D1 combinations.

The task ID1 $\rightarrow$ D1 represents the "Sussman anomaly" [201, 227, 230, 137]. The problem is considered an anomaly because you cannot do either of the necessary first actions without undoing it at a later point. If you first put block A on block B after putting block C on the table, you will have to take it off again so that you can move block B onto block C. If you put block B on block C first, you will have to undo this to move block A. This is also what the associative computer using the $h2$ heuristic does (see fig. 6.7).

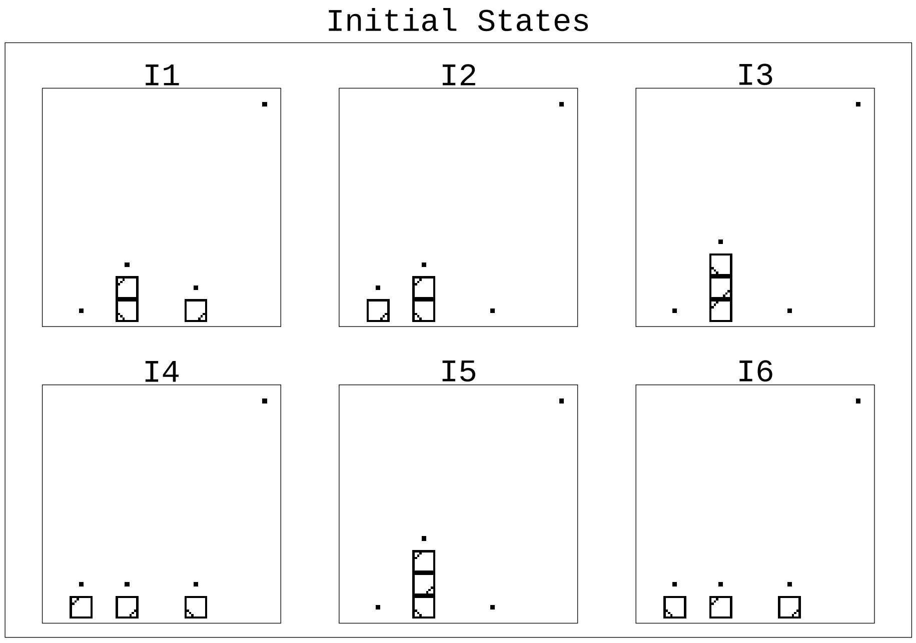


Figure 6.5: Six initial states.

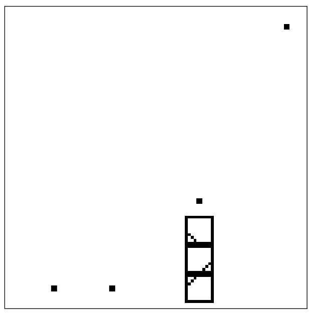


Figure 6.6: The Desired state, CBA tower.

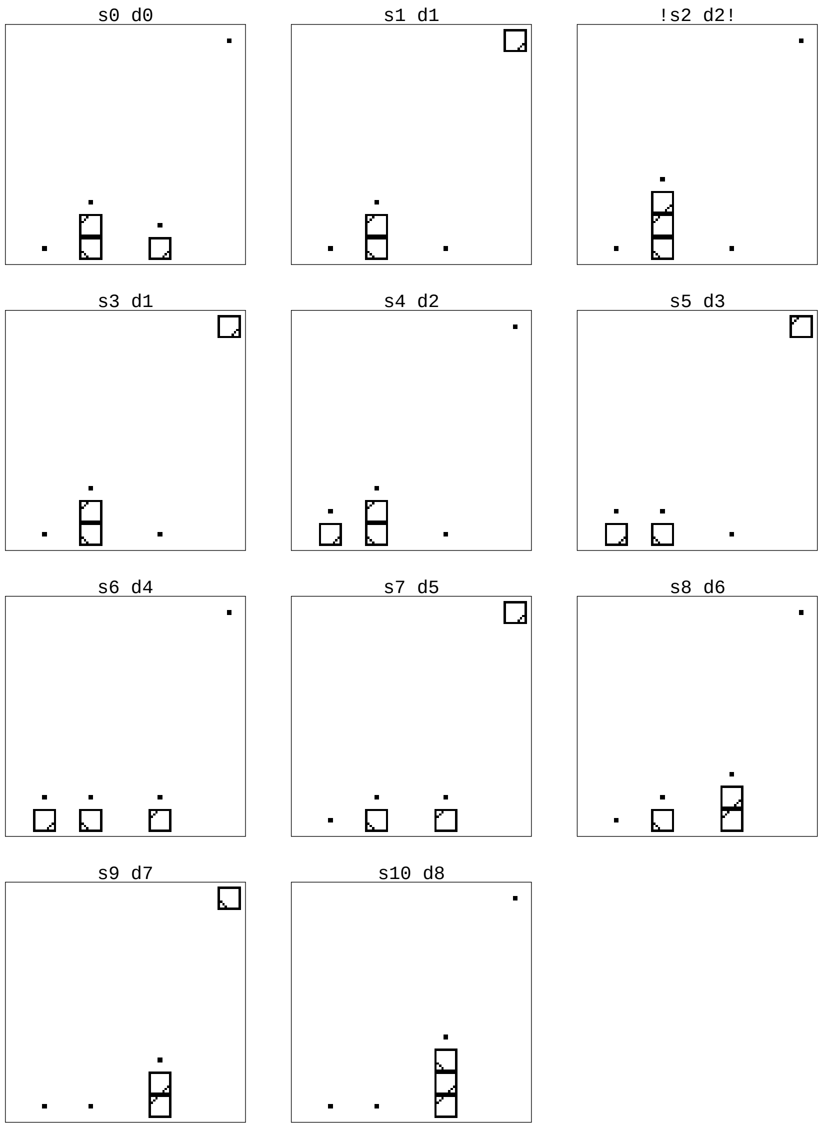


Figure 6.7: Planning of the task ID1 → D1 ("Sussman anomaly") with heuristic h2.

6.1.3 Tower, few initial states

Supervised learning

In this experiment we examined how to build a tower CBA (fig. 6.6). 59 different initial states are possible. Six initial states which covered the problem space of building the CBA tower (fig. 6.5) were chosen. The steps, plan length and backtracking steps of the six tasks were determined using $h0, h1$, and $h2$ as shown in tables 6.4, 6.5, 6.6. After that, two learning sessions were performed[1]. The $h2$ heuristic guided the search. In the l learning session, the Sussman anomaly example ID1 → D1 was learned. In the *all* learning session all 6 examples were learned. The plan length for the prediction heuristic *all* was the same as for the pattern heuristic $h2$, but no backtracking steps occurred. The prediction associative memory which learned the Sussman anomaly did not have enough knowledge to significantly improve the search.

task	h0	h1	h2	l	all
I1 → D1	94	32	10	8	8
I2 → D1	86	20	20	6	16
I3 → D1	12	22	22	46	18
I4 → D1	90	26	26	26	20
I5 → D1	66	6	6	28	6
I6 → D1	40	48	18	48	14
mean	64.67	25.67	17	27	13.67
sdev	32.68	13.94	7.56	17.92	5.57
p	-	**0.0252**	**0.0101**	0.0594	**0.0075**

Table 6.4: Required steps, build CBA tower task.

task	h0	h1	h2	l	all
I1 → D1	50	24	8	8	8
I2 → D1	42	16	16	6	16
I3 → D1	10	18	18	32	18
I4 → D1	46	20	20	26	20
I5 → D1	46	6	6	26	6
I6 → D1	32	30	14	38	14
mean	37.67	19	13.67	22.67	13.67
sdev	14.88	8.08	5.57	12.94	5.57
p	-	**0.0256**	**0.0115**	0.0979	**0.0115**

Table 6.5: Plan length, build CBA tower task.

[1]The add of the desired state to the question vector of the prediction associative memory was not necessary. However, for compatibility reasons, it was performed.

task	h0	h1	h2	l	all
I1 → D1	22	4	1	0	0
I2 → D1	22	2	2	0	0
I3 → D1	1	2	2	7	0
I4 → D1	22	3	3	0	0
I5 → D1	10	0	0	1	0
I6 → D1	4	9	2	5	0
mean	13.5	3.33	1.67	2.17	0
sdev	9.7519	3.0768	1.0328	3.06	0

Table 6.6: Backtracking steps, build CBA tower task.

Unsupervised learning

The prediction associative memory was empty at the beginning of the unsupervised learning. The unsupervised learning brought no significant improvement due to the interference of the problem I4 → D1 with other problems (see fig. 6.8). The equilibrium was already reached at step two.

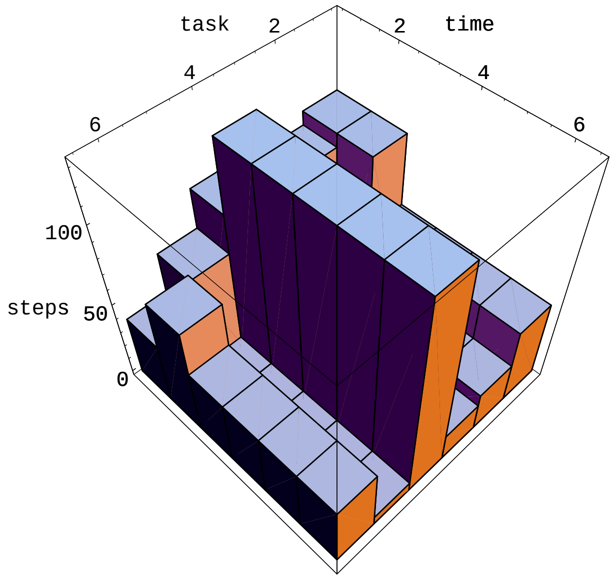


Figure 6.8: Required steps, build CBA tower task during learning.

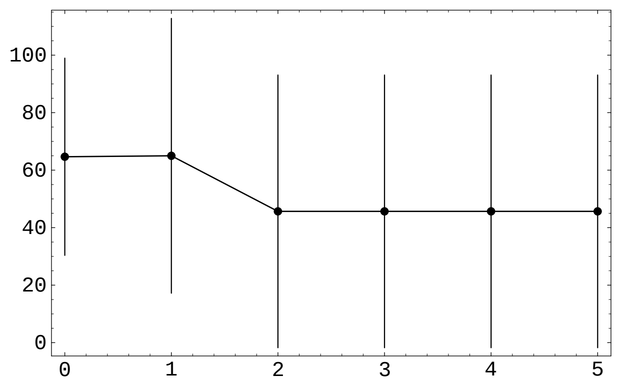


Figure 6.9: Mean steps for learning. *t2-t20: mean=45.67, sdev=45.19, p=0.16*

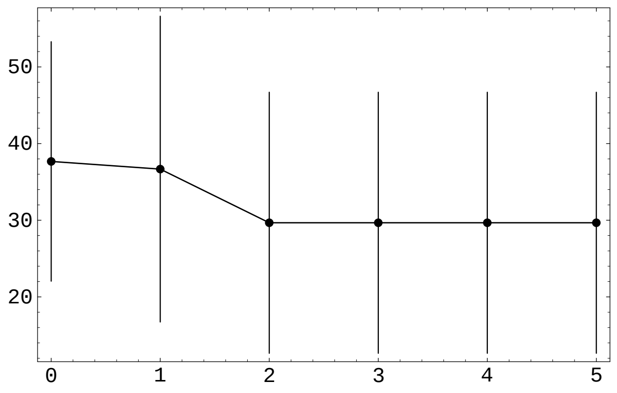


Figure 6.10: Mean plan length for learning. *t2-t20: mean=29.67, sdev=16.22*

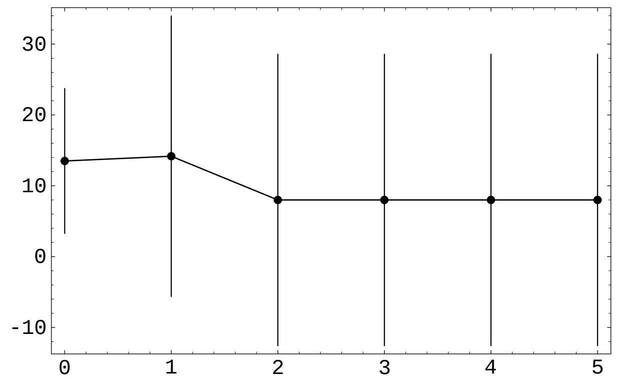


Figure 6.11: Mean backtracking steps for learning. *t2-t20: mean=8, sdev=19.6*

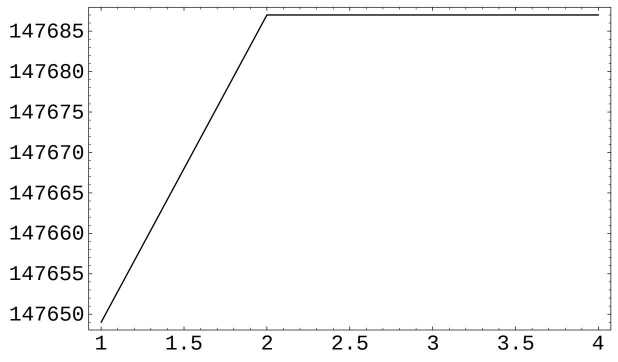


Figure 6.12: The number of synapses which are not zero during learning. *t2-t20: 147687 of* 1.99101^8 *synapses are not zero (0.0742%).*

Context information resolved the problem of the interference of the problem I4 → D1 with other problems (see fig. 6.8). By detachment of the particular solution, significant improvement in the required steps and the plan length was obtained. The equilibrium of the associative computer was reached one stage after the stabilization in the behavior of the prediction associative memory, at stage three (see fig. 6.17).

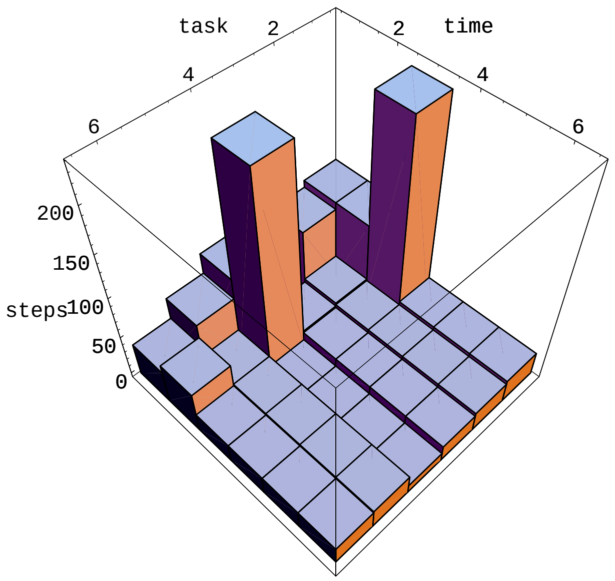


Figure 6.13: Required steps, small build CBA tower task with context during learning.

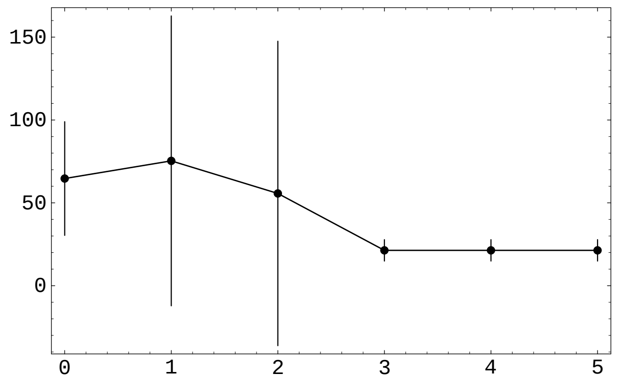


Figure 6.14: Mean steps for learning. *t3-t20: mean=21.33, sdev=6.15,* ***p=0.00119***

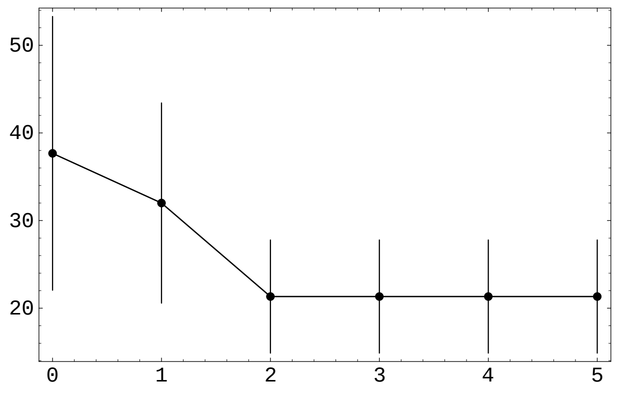


Figure 6.15: Mean plan length for learning. *t2-t20: mean=21.33, sdev=6.15,* ***p=0.0242***

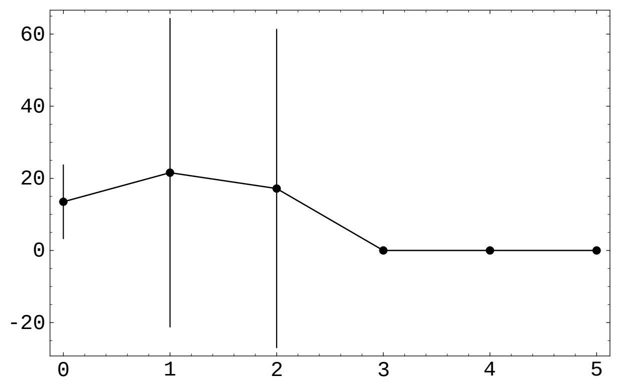


Figure 6.16: Mean backtracking steps for learning. *t3-t20: mean=0, sdev=0*

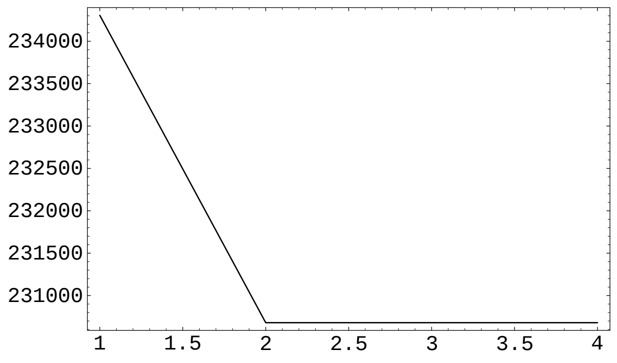


Figure 6.17: The number of synapses which are not zero during learning. *t2-t20: 230680 of* 2.98801^8 *synapses are not zero (0.0772%).*

Destroy tower

In this task the desired state (fig. 6.6) became the initial state. The former initial states (fig. 6.5) became the desired states. While the initial problem is inverse, the computed solutions are not. Neither the pattern heuristic nor the prediction heuristic, which was supervised learned, brought any significant improvement (see table 6.7). The computation by the associative computer is not symmetric to problems.

task	h0	h2	all
D1 → I1	30	12	10
D1 → I2	60	24	22
D1 → I3	12	22	18
D1 → I4	22	10	10
D1 → I5	66	6	6
D1 → I6	10	40	32
mean	33.33	19	16.33
sdev	24.16	12.44	9.67
p	-	0.16	0.11

Table 6.7: Required steps. The tasks in italic are symmetric in both directions, the others are not.

The unsupervised learning brought a significant improvement[2]. The back-tracking steps were eliminated (see fig. 6.21).

[2]Because the initial state was always the same, context information was of no importance.

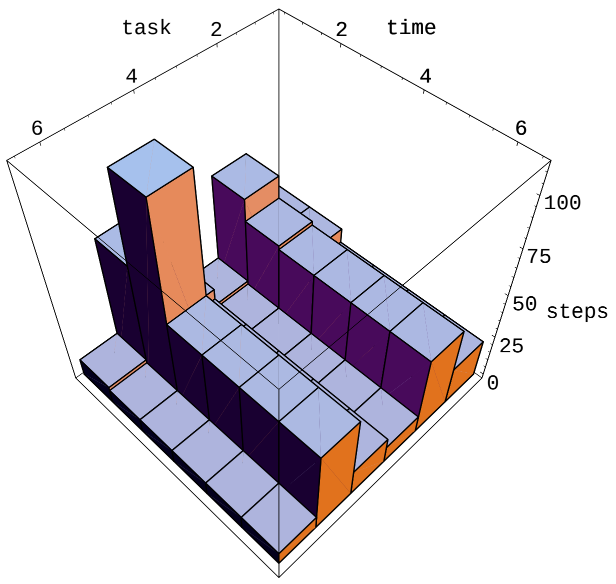


Figure 6.18: Required steps, destroy CBA tower task during learning.

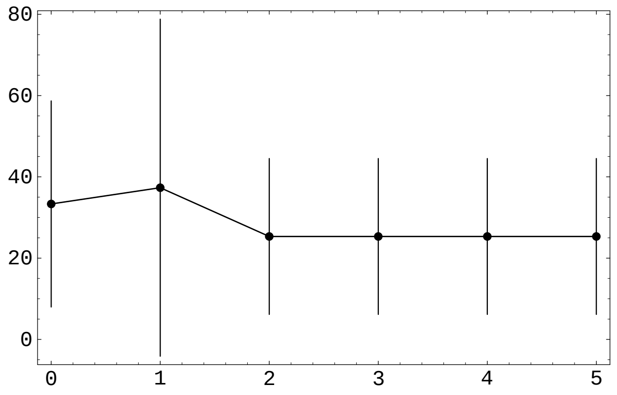


Figure 6.19: Mean steps for learning. *t2-t10: mean=25.33, sdev=18.27,* ***p=0.01***

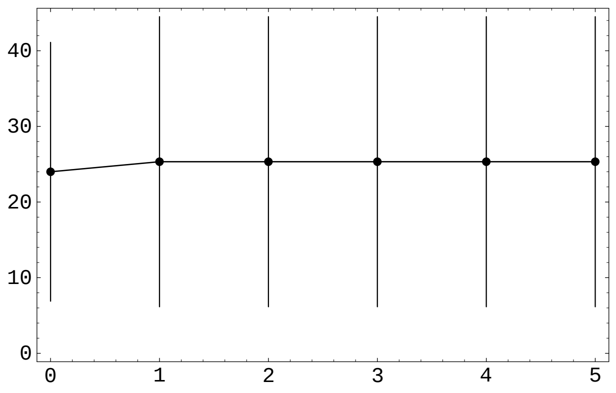


Figure 6.20: Mean plan length for learning. *t1-t20: mean=25.33, sdev=18.27*

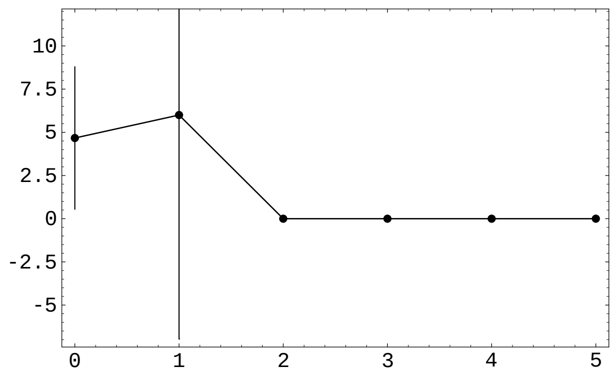


Figure 6.21: Mean backtracking steps for learning. *t2-t20: mean=0, sdev=0*

6.1.4 Tower, most important initial states

Of the remaining 53 possible states, 28 initial states were chosen and the remaining 25 states were not considered. There were two reasons for not including a state. First, some of the states represent trival problems, for example only one move from the desired state. The other reason was that some states were already nearly described by other states, for example different combinations of towers in the middle position, versus different combinations of towers in the left position (see fig. 6.22-6.26).

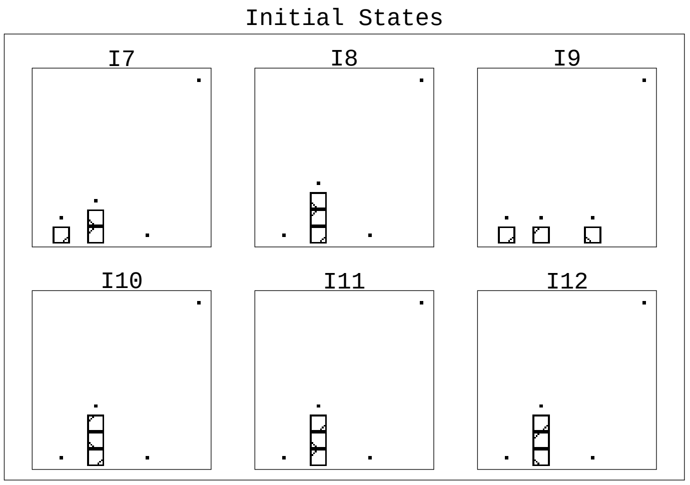


Figure 6.22: The remaining initial states I7-I12 for the task CBA tower.

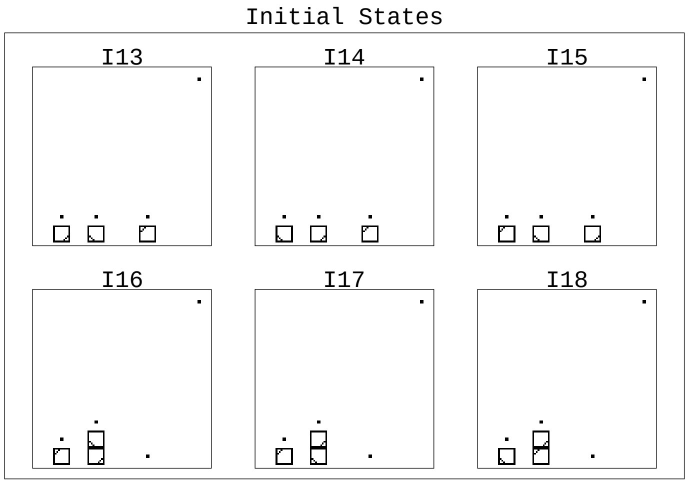


Figure 6.23: The remaining initial states I13-I18 for the task CBA tower.

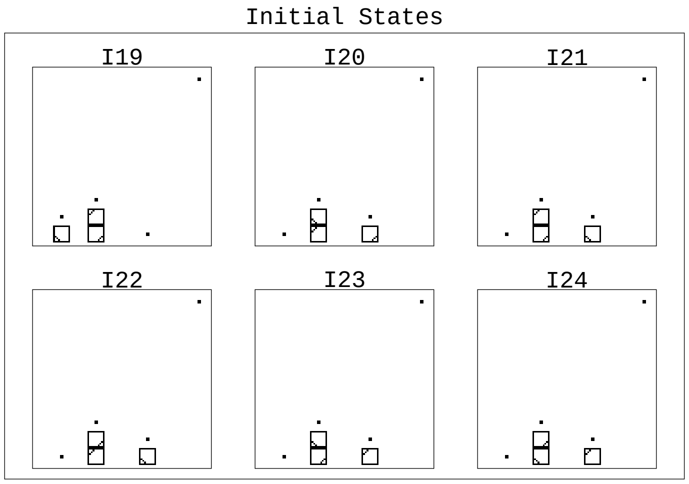


Figure 6.24: The remaining initial states I19-I24 for the task CBA tower.

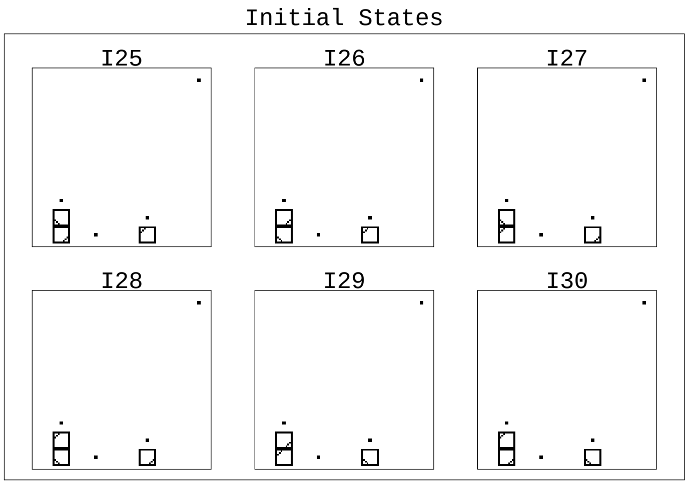


Figure 6.25: The remaining initial states I25-I30 for the task CBA tower.

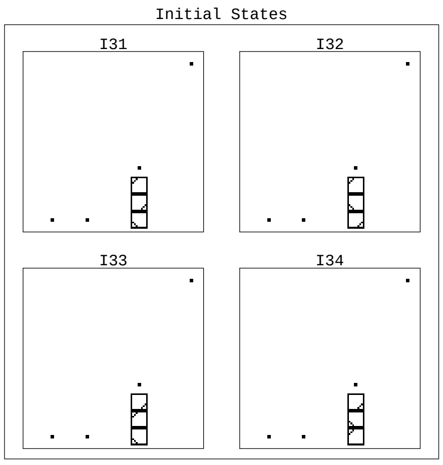


Figure 6.26: The remaining initial states I31-I34 for the task CBA tower.

Generalization

The 28 initial states were used to test the generalization of the prediction associative memory trained by the first six examples (see table 6.4). The prediction associative memory which was formed by supervised learning of the six examples (here labeled as all^{small}) produced the best results (see tables 6.8, 6.9, 6.10). No backtracking steps were performed. The example ID31 → D1 represents the task of the rebuilding of the tower (see fig. 6.27). The best possible solution was found using the prediction heuristic all^{small}. The six examples are sufficient to characterize the problem of building the CBA tower. The prediction associative memory which was formed by learning the Sussman anomaly ID1 → D1 describes the problem of building the CBA tower statistically better than the pattern heuristic. The prediction associative memory which was formed by the unsupervised learning (us) of the six examples (see fig. 6.9) produced weaker results than the pattern heuristics. Context unsupervised learning of the six examples brought significant improvement for those examples (see fig. 6.14). However, it failed to generalize and it was weaker than the unsupervised learning. Context can help by the detachment of particular solutions, however, it should be not used if generalization is required.

	h0	h1	h2	l	all^{small}	us^{small}	cus^{small}
mean	61.43	32.57	28.21	24.29	11.21	35.89	42.43
sdev	34.73	27.01	19.36	16.4	5.67	30.08	38.67
p	-	0.00019	0.000086	6.7^{-6}	1.28^{-8}	0.0015	0.012

Table 6.8: Required steps for build tower I7-I34 → D1.

	h0	h1	h2	l	all^{small}	us^{small}	cus^{small}
mean	37.5	22.57	21.64	20.5	11.21	24.93	24.86
sdev	15.08	15.49	13.97	10.88	5.67	12.41	12.21
p	-	0.000094	0.00015	0.000018	1.76^{-9}	0.00038	0.00073

Table 6.9: Required plan length for build tower I7-I34 → D1.

	h0	h1	h2	l	all^{small}	us^{small}	cus^{small}
mean	11.96	5	3.29	1.89	0	5.46	8.79
sdev	11.45	6.57	2.72	3.39	0	10.78	15.01
p	-	0.0013	0.0003	0.000045	3.7^{-6}	0.012	0.145

Table 6.10: Required backtracking steps for build tower I7-I34 → D1.

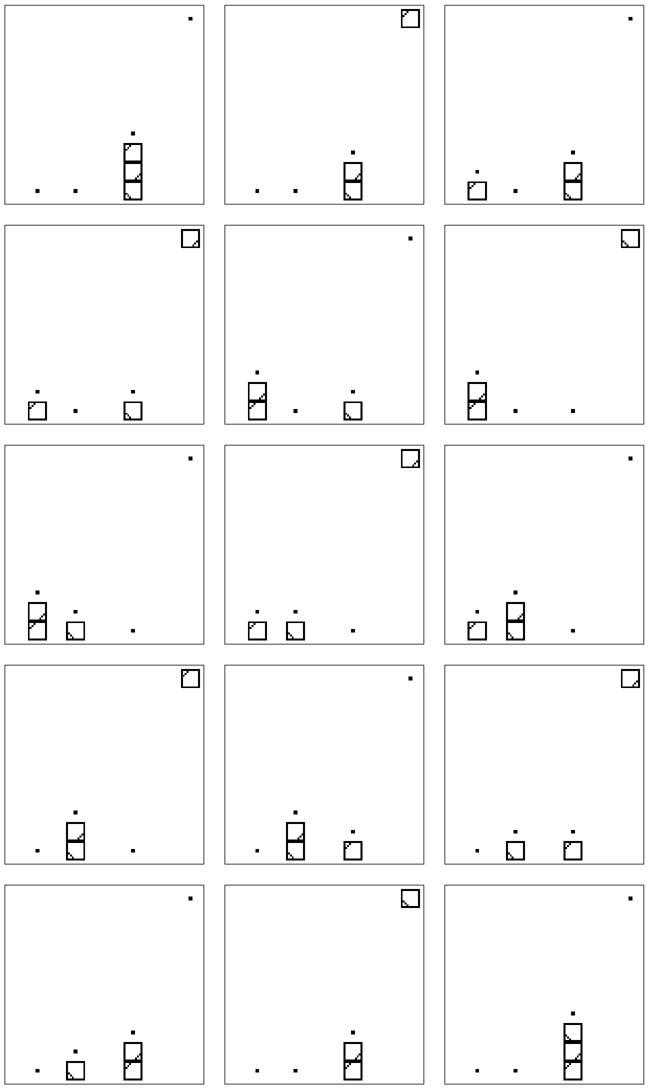


Figure 6.27: Planning of the task ID31 → D1 after all^{small} learning. The best possible solution was found.

Again it is shown how important is the choice of the appropriate examples as teaching material. The prediction associative memory which was formed by learning the Sussman anomaly describes the problem of building the CBA tower statistically better than the pattern heuristic. This is really amazing. One well-chosen example catches all the essential information about the nature of building the CBA tower.

Supervised learning

All 34 examples were learned by the unsupervised learning (noted *all*). The search was guided by the $h2$ heuristic. The results were comparable to the $h1$ pattern heuristic (see tables 6.11, 6.12, 6.13). These weak results were due to the overlearning of the prediction associative memory. Beside the overloading of the prediction associative memory, the interference between different problems also had an effect. Because of this, the added context improved the results (noted as c-all). But the quality is not comparable to the results obtained by the learned six examples, or the Sussman anomaly. The conclusion: the prediction associative memory can overlearn.

	h0	h1	h2	l	all^{small}	all	c-all
mean	62	31.35	26.24	24.77	11.65	33	31.71
sdev	33.92	25.17	18.28	16.43	5.65	25.136	21.79
p	-	0.00002	3.79^{-6}	1.44^{-6}	2.44^{-10}	0.000027	0.00005

Table 6.11: Required steps for build tower I1-I34 → D1.

	h0	h1	h2	l	all^{small}	all	c-all
mean	37.53	21.94	20.24	20.88	11.65	26	27
sdev	14.82	14.43	13.19	11.09	5.65	13.19	14.98
p	-	7.94^{-6}	7.51^{-6}	7.24^{-6}	5.55^{-11}	0.0003	0.0017

Table 6.12: Required plan length for build tower I1-I34 → D1.

	h0	h1	h2	l	all^{small}	all	c-all
mean	12.24	4.71	3	1.94	0	3.38	2.35
sdev	11.05	6.09	2.57	3.29	0	6.28	5.58
p	-	0.0002	0.00002	5.52^{-6}	1.26^{-7}	0.00003	0.00005

Table 6.13: Required backtracking steps for build tower I1-I34 → D1.

Unsupervised learning

The prediction associative memory was examined during unsupervised learning of many problems. The computer resources were restricted to a quantity of 600 steps. At the beginning of the learning the analogy memory was empty. A stage consists of the tasks I1 → D1, I2 → D2,.., I34 → D1. At the third time interval, third stage, a strong interference of the task I17 → D1 with other tasks occurred (see fig. 6.28). Despite the determined arrangement of the tasks, the interference of the task I17 → D1 could be eliminated at time interval 10. The elimination can be recognized in the graphs representing the required steps, plan length and the required backtracking steps (see fig. 6.29, 6.30, 6.31). It is seen most clearly in the figure representing the number of synapses of the prediction associative memory which are not zero (see fig 6.32). The equilibrium of the behavior of the associative computer was reached at step 14. The results are not persuasive, despite the significant improvement to the blind search $h0$.

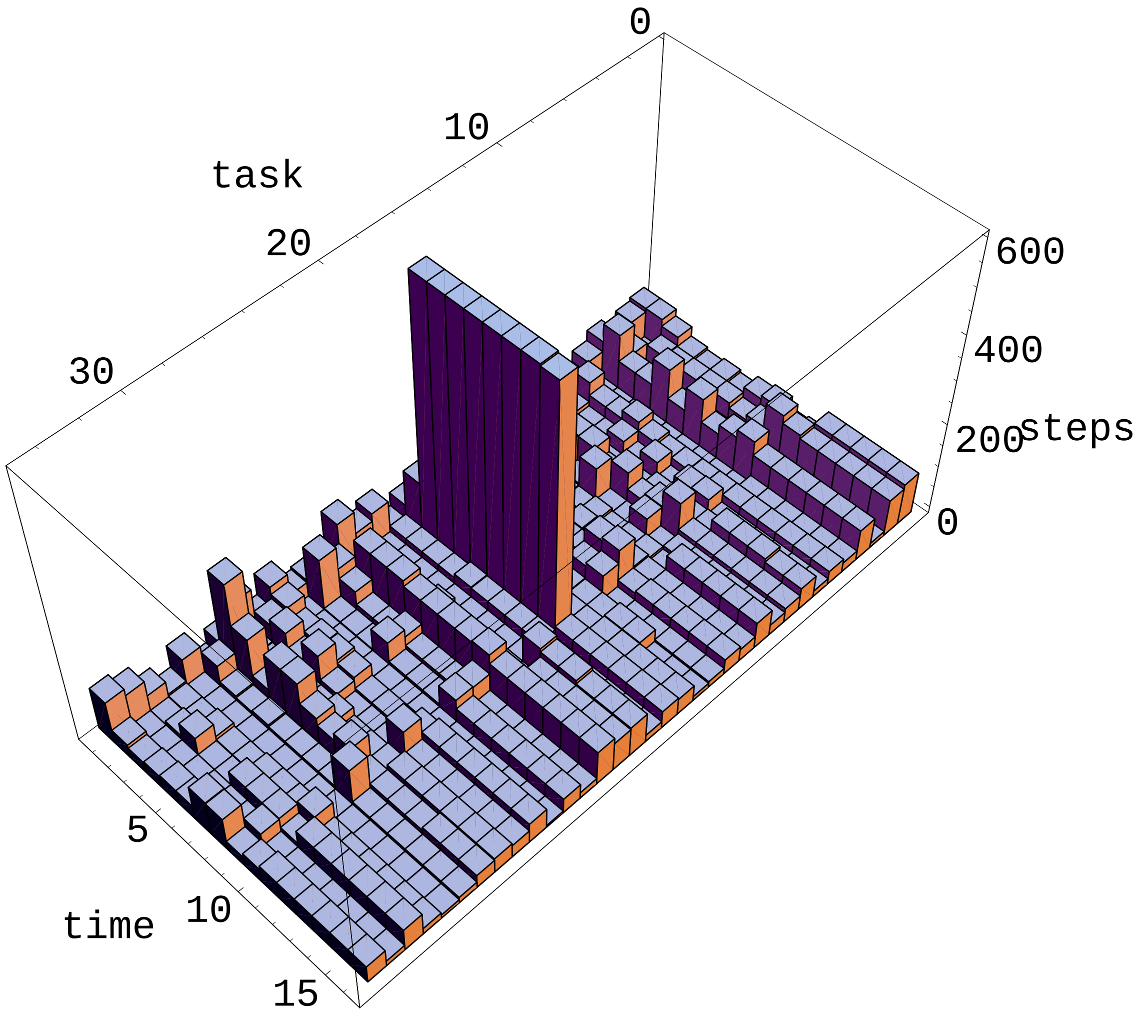


Figure 6.28: Required steps, build CBA tower during learning.

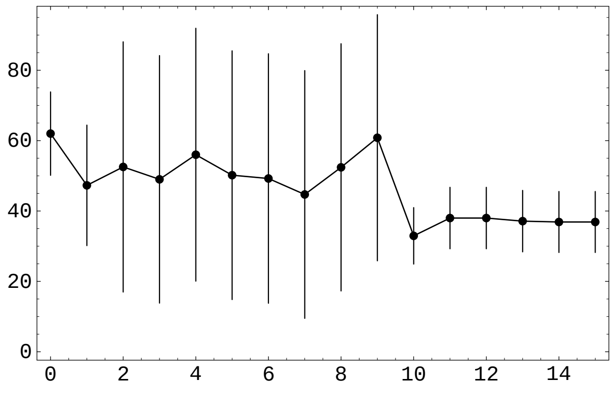


Figure 6.29: Mean steps for learning. *t14-t15: mean=36.88, sdev=24.81,* ***p=0.0019***

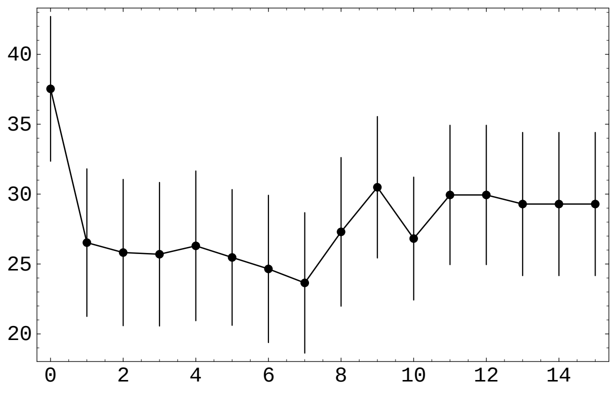


Figure 6.30: Mean plan length for learning. *t14-t15: mean=29.29, sdev=14.66,* ***p=0.00001***

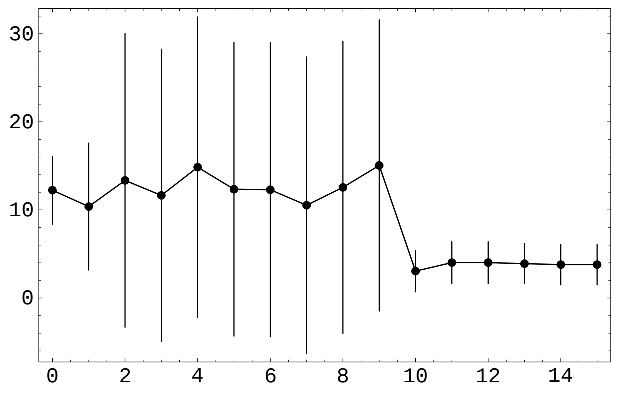


Figure 6.31: Mean backtracking steps for learning. *t14-t15: mean=3.79, sdev=6.6,* ***p=0.000012***

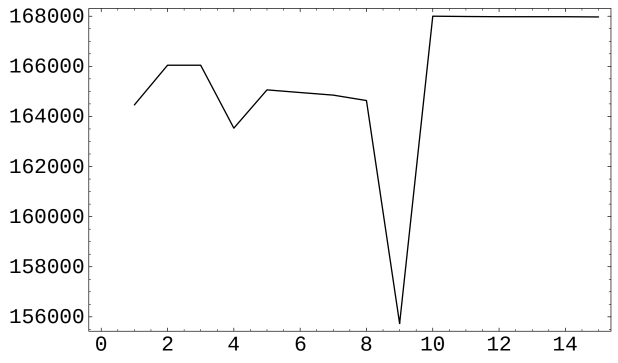


Figure 6.32: The number of synapses which are not zero during learning. *t15: 167971 of* 1.99101^8 *synapses are not zero (0.0844%).*

Context information resolved the problem of the interference. The results are obviously better (see fig. 6.33, 6.34, 6.35, 6.36). They are comparable to the supervised learning *all*, which was guided by the heuristic function $h2$. The equilibrium of the behavior of the associative computer was reached at the time interval eight. However, the equilibrium of the prediction associative memory was reached at interval eleven (see fig. 6.37)

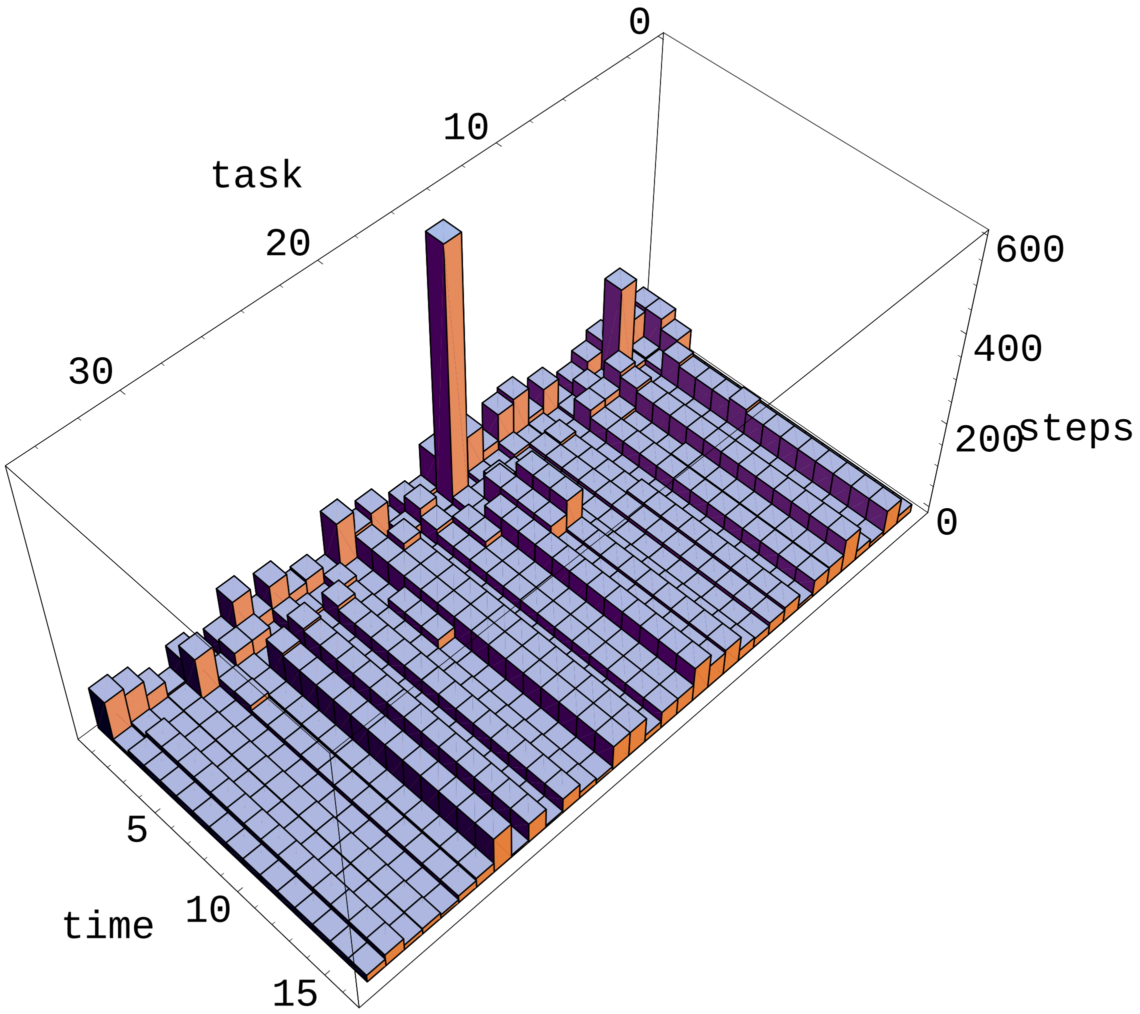


Figure 6.33: Required steps, build CBA tower with context during learning.

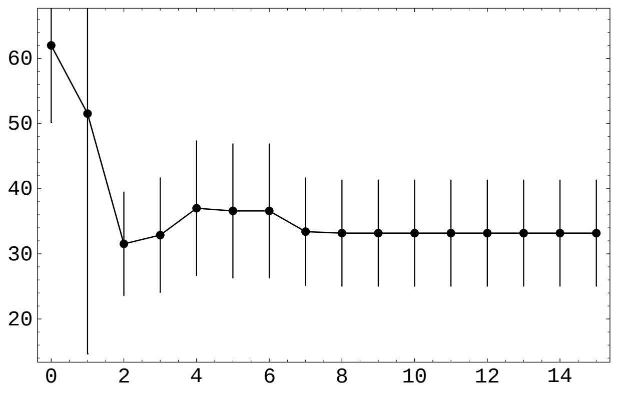


Figure 6.34: Mean steps for learning. *t8-t15: mean=33.18, sdev=23.31,* ***p***$=9.93^{-6}$

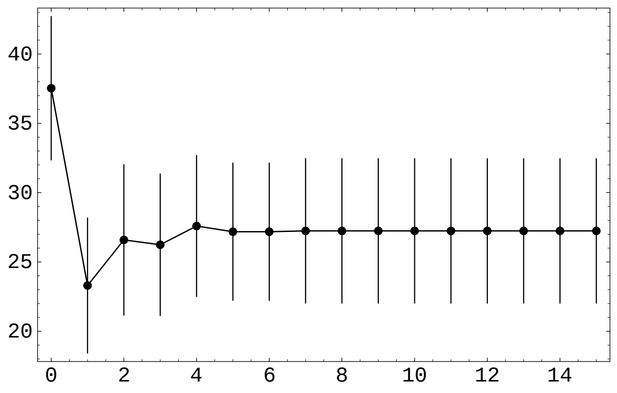


Figure 6.35: Mean plan length for learning. *t7-t15: mean=27.24, sdev=14.92,* ***p=0.0016***

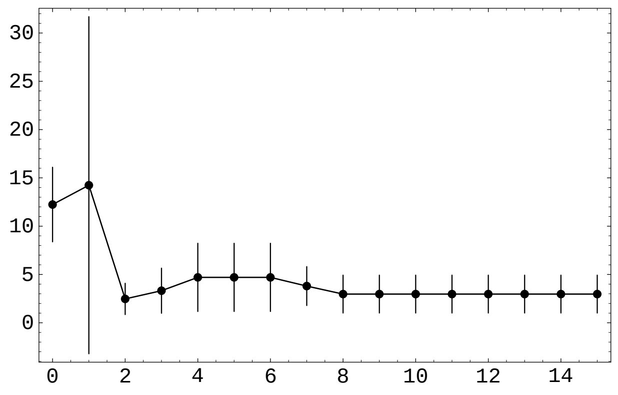


Figure 6.36: Mean backtracking steps for learning. *t8-t15: mean=2.97, sdev=5.61,* $\boldsymbol{p}$=$9.55^{-}7$

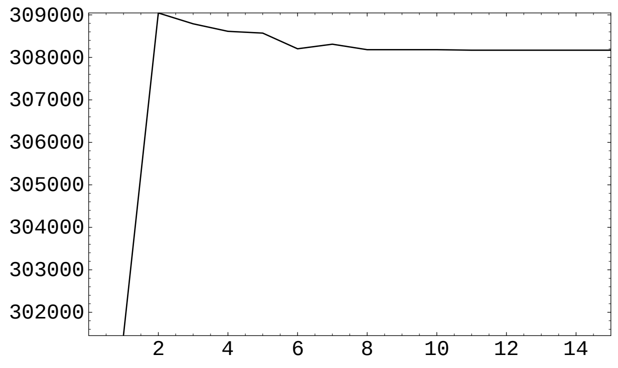


Figure 6.37: The number of synapses which are not zero during learning. *t11-t15: 308169 of* 2.98801^{8} *synapses are not zero (0.1031%).*

In this example it was shown how unsupervised learning of many problems is performed. Many states which differ only slightly describe different problems. Despite this fact, the prediction associative memory extracts the relevant knowledge. Because of this, many more learning sequences are needed until the stabilization of the behavior of the associative computer. But at least there is a stabilization to which the model converges. After the first learning sessions, an impairment is present. The improvement comes along with the reduction of the weights of the prediction associative memory. The number of required learning steps (sequences) until an equilibrium in the behavior is reached is proportional to the number of problems which are learned. Many problems supply many contradictory knowledge to the prediction associative memory. This becomes clear in the experiment where context led to a faster improved learning. Additional context such as modulation should bring even better improvement.

Observations

Supervised learning with an observer who had chosen relevant examples led to the best results (all^{small}, Sussman anomaly). The failure of the teacher to chose relevant examples led to an impairment. The results obtained after the learning of all 34 problems were worse than the $h2$ pattern heuristic, despite the fact that the $h2$ heuristic guided the search. This is due to overlearning of the prediction associative memory. Unsupervised learning led to weak generalization performance[3]. Unsupervised learning of all problems, particularly with context, was only slightly worse than the pattern heuristic.

Prediction associative memory

The supervised all^{small} learning strategy led to the best results. The prediction associative memory is composed of 22 building blocks of columns and rows. The distribution represented a zigzag with six frequency maxima which lay over the value 20 (see figures 6.38). The load of the prediction associative memory which resulted from the supervised learning of all examples *all* is higher. It is composed of 22 building blocks of columns, and only 20 of rows. The number of the frequency maxima which lay over 20 is reduced to five (see figures 6.39). The load of the prediction associative memory which resulted from the unsupervised learning of all examples without context is higher than the preceding examples. It is, however, composed of even fewer building blocks (19 building blocks of columns and 17 of rows). The number of the frequency maxima which lay over 20 is four (see figures 6.40). The gain of load with the reduction of blocks indicates the loss of important information. This information distinguishes the different states and is concentrated in small areas of the weight matrix of the prediction associative memory.

[3]Nevertheless the improvement is significant.

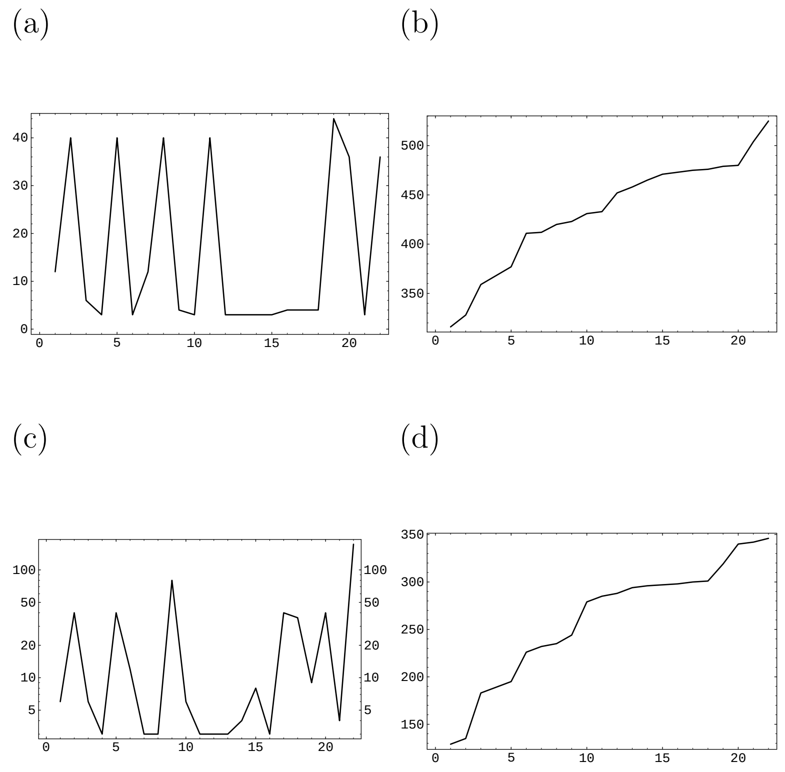
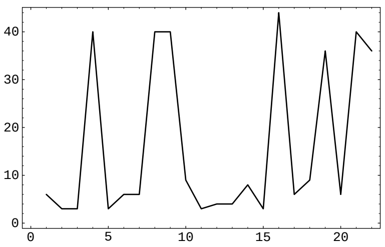
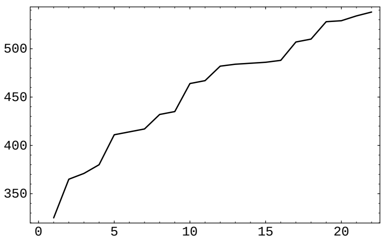


Figure 6.38: 0.0746% synapses are not zero of associative memory of the task *all*^{small}. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows *(logarithmic plot).* (d) The corresponding sum values of rows.

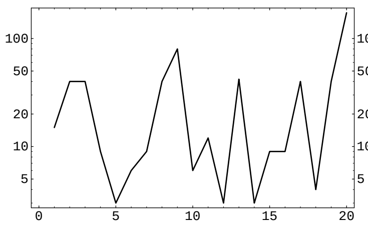
(a)



(b)



(c)



(d)

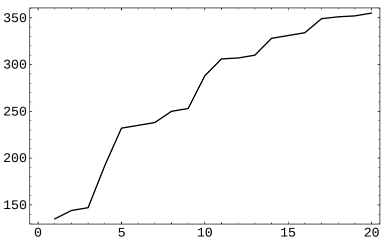


Figure 6.39: 0.0839% synapses are not zero of associative memory of the task *all*. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows *(logarithmic plot).* (d) The corresponding sum values of rows.

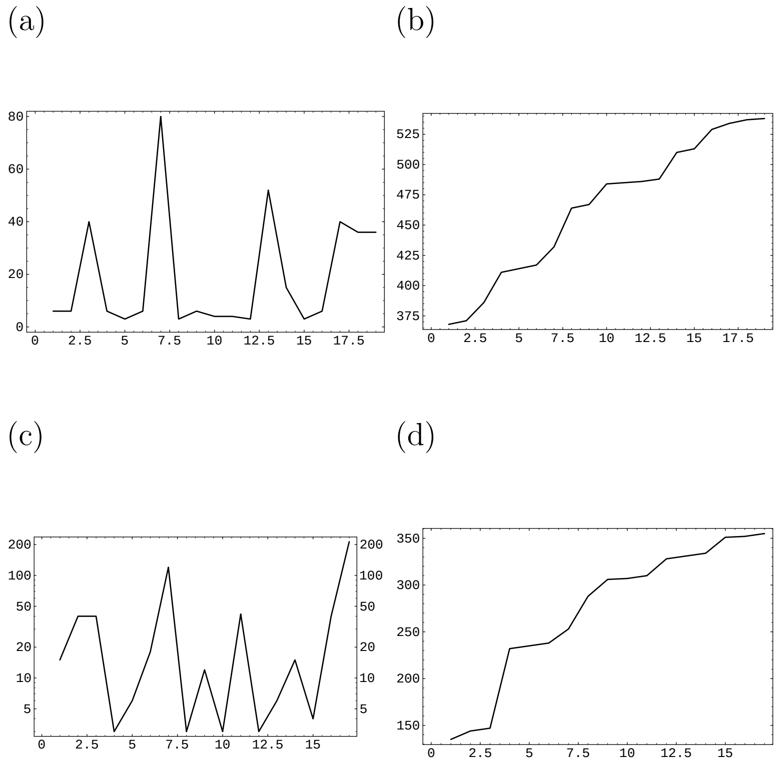


Figure 6.40: 0.0844% synapses are not zero of associative memory of the task unsupervised learning of 34 examples. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows *(logarithmic plot).* (d) The corresponding sum values of rows.

Context information increased the load and the number of the building blocks of the prediction associative memory (see figures 6.41 and 6.42). The number of the maxima remained the same as without the context.

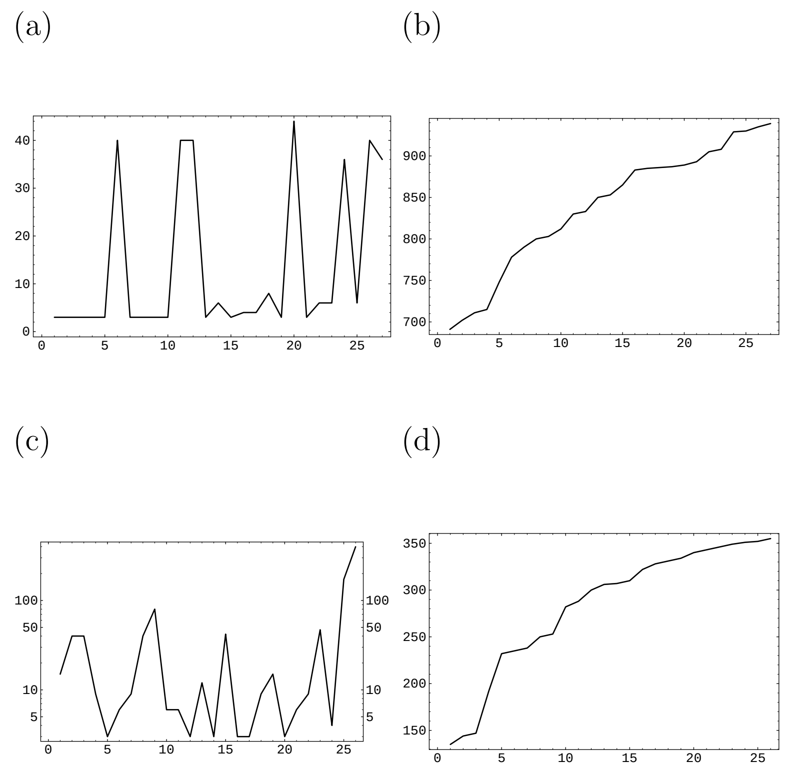


Figure 6.41: 0.1549% synapses are not zero of associative memory of the task *c-all*. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows *(logarithmic plot).* (d) The corresponding sum values of rows.

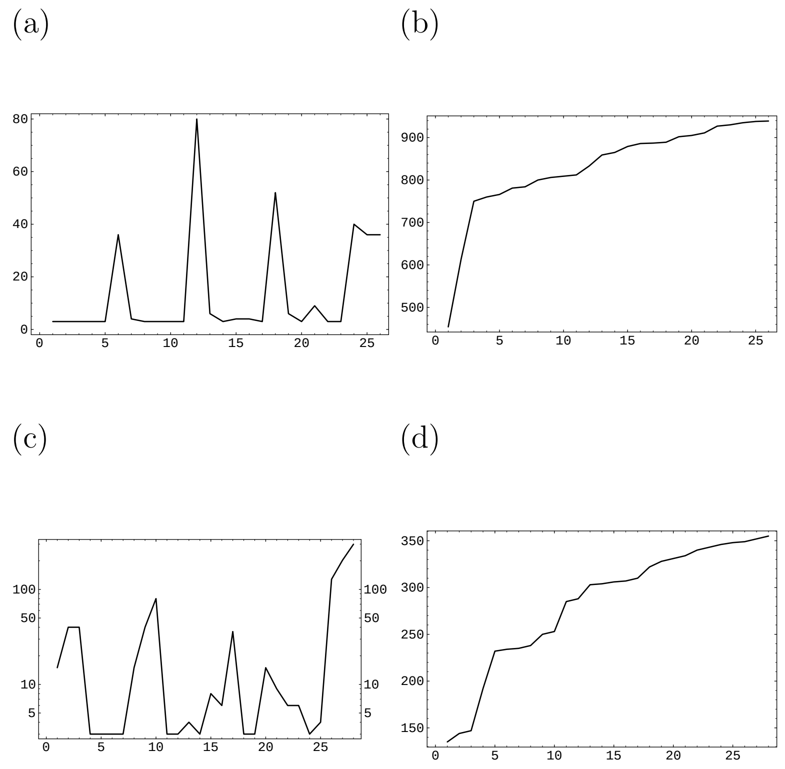


Figure 6.42: 0.1031% synapses are not zero of associative memory of the task unsupervised learning of 34 examples with context. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values rows *(logarithmic plot)*. (d) The corresponding sum values of rows.

6.1.5 Additional blocks

The integration of the associative computer into bigger systems, such as intelligent planning systems, requires the representation of additional objects. This possibility is examined in the next experiment. Six additional different blocks are introduced. The blocks differ by different marks in the counter (see fig 6.43). 378 additional associations were learned by the permutation associative memory. These associations represent the additional blocks and additional positions resulting from stacking the blocks one upon the other.

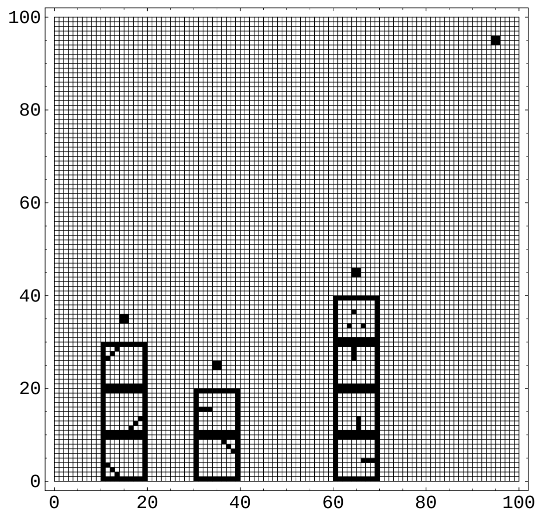


Figure 6.43: The world for the task 9 blocks.

The differing positions correspond to the coding of the blocks and their vertical positions. For blocks representation positions 1 to 100, 301 to 400 and 601 to 700 correspond. For vertical positions 101 to 200 and 401 to 500 correspond (see fig. 6.44).

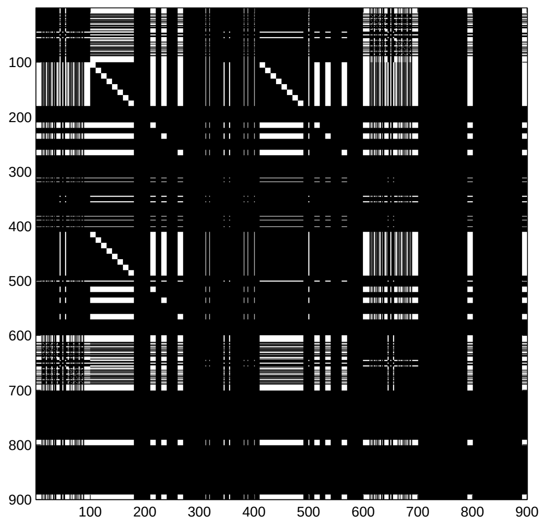


Figure 6.44: The weight matrix of the permutation associative memory for the task 9 blocks. 89788 synapses are not zero, 11.0849% of the weight matrix.

The weight matrix is composed of eleven elementary blocks (see fig 6.45). Two frequency maxima and two frequency minima blocks remained as before the additional learning (see fig 6.3).

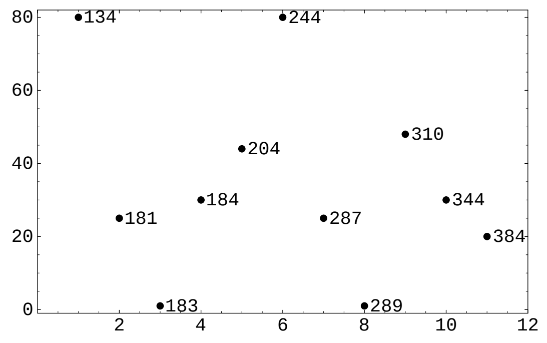


Figure 6.45: The sum of the synapses of a column. The x axis indicates the frequency of different sum values of columns. The written number corresponds to the sum values.

Supervised learning

In the example, 6 tasks were solved with the aid of the pattern heuristic $h2$. Those tasks were also learned by the prediction associative memory. They were recalled without error. The 6 tasks correspond to the decomposition of the problem I1 $\rightarrow$ D1.

task	h2	*all*
I1 $\rightarrow$ ID2	4	4
ID2 $\rightarrow$ ID3	4	4
ID3 $\rightarrow$ ID4	4	4
ID4 $\rightarrow$ ID5	2	2
ID5 $\rightarrow$ ID6	6	6
ID6 $\rightarrow$ D1	4	4

Table 6.14: 100 percent recognition after one step learning for 6 transitions.

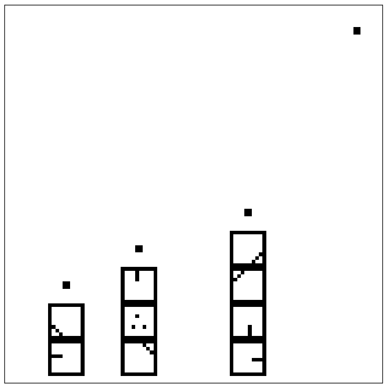


Figure 6.46: The desired state D1 for the task 9 blocks.

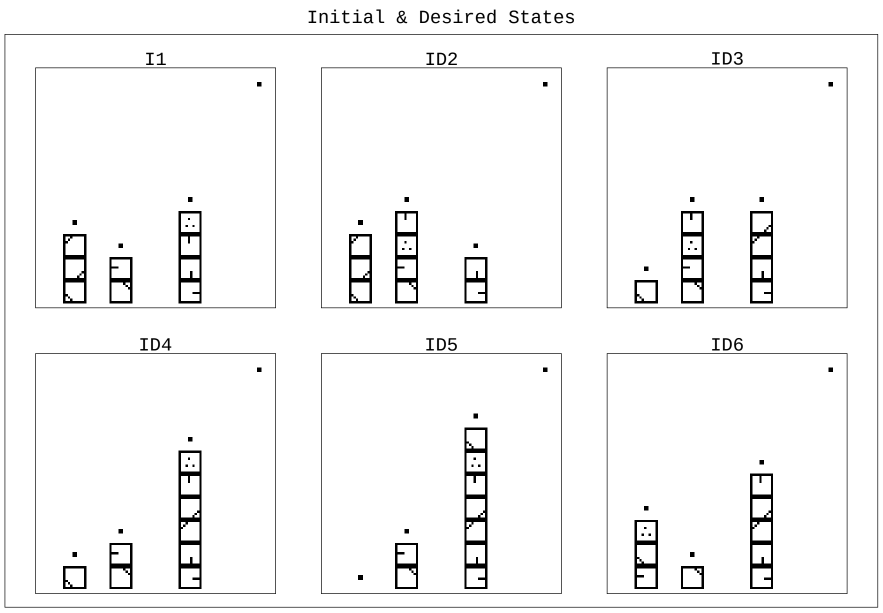


Figure 6.47: The initial and desired states for the task 9 blocks.

6.2 8-Puzzle

The 8-puzzle is composed of eight numbered movable tiles in a 3×3 frame. One cell of the frame is empty and because of this, tiles can be moved around to form different patterns. The goal is to find a series of moves of tiles into the blank space that changes the board from the initial configuration to a desired configuration (see fig. 6.48). Two common heuristics for this task are the number of misplaced tiles, and the "city-block distance" [136, 151, 111]. The first heuristic counts the number of misplaced tiles out of place in each state compared to the desired goal. However this heuristic fails to take into account all available information such as the distance the tiles must be moved. The "city-block distance" sums all the distances by which the tiles are out of place, with one count for each square a tile must be moved to reach a position of the desired state. The "city-block distance", also called the "Manhattan distance", is often better than the "number of misplaced tiles".

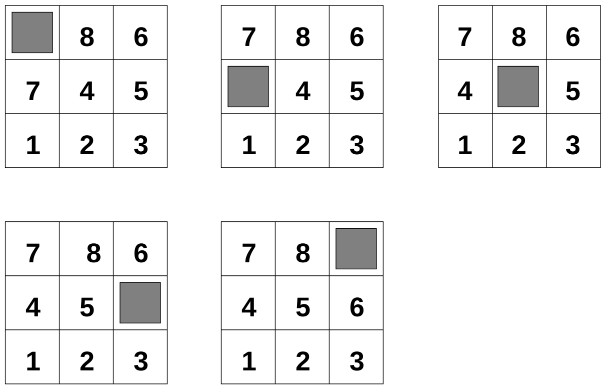


Figure 6.48: The first pattern (upper left) represents the initial configuration and the last (low right) the desired configuration. The series of moves describe the solution to the problem.

6.2.1 Pattern representation of the 8-puzzle

The Oksapmin tribe of Papua New Guinea counts by associating a number with the position of the body [103]. This suggests a representation of numbers by bars at certain positions which can overlap. A bar at a certain position codes the magnitude of the number. The closeness or similarity of different numbers is determined by the overlap of the bar codes (see fig. 6.49).

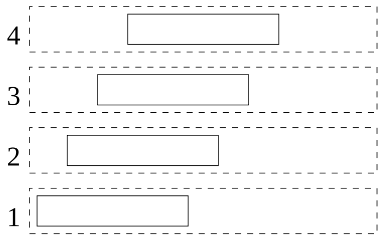
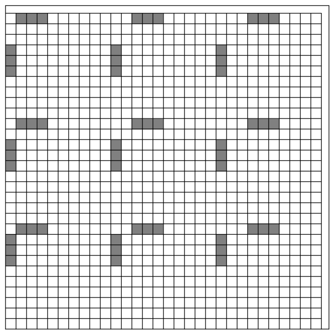


Figure 6.49: A bar at a certain position codes the magnitude of the number. The closeness or similarity of different numbers is determined by the overlap of the bar codes.

In the 8-puzzle, each tile is defined by its corresponding coordinates. Two numbers can be represented by two bars (see fig 6.50). The amount of overlapping indicates the closeness of different tiles.



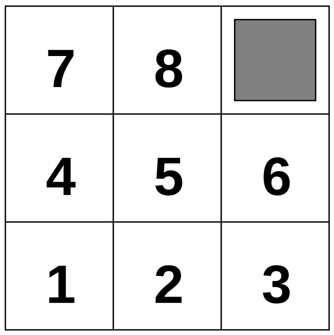


Figure 6.50: The desired state for the task 8-puzzle and its representation by bars. The associative fields in which the objects are describe have a fixed dimension of ten times ten pixels. Because of this, excessive unused space is present.

A tile is represented by a cognitive entity. The tile is specified by the first associative field of dimension ten times ten, its positions by the succeeding associative fields. Bars have the size of ten because of compatibility and clarity. The desired state is represented by nine cognitive entities:

```
No:1

0111000000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

No:2

0011100000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:3

0001110000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000

No:4

0111000000
0000000000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
```

```
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

No:5

0011100000
0000000000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:6

0001110000
0000000000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000

No:7

0111000000
0000000000
0000000000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

No:8

0011100000
0000000000
0000000000
1000000000
1000000000
```

```
1000000000
0000000000
0000000000
0000000000
0000000000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:9

0001110000
0000000000
0000000000
1000000000
1000000000
1000000000
0000000000
0000000000
0000000000
0000000000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000
```

6.2.2 Permutation associative memory

192 associations were learned. Its premises and conditions are described by two cognitive entities (see fig 6.51). They describe all possible moves of the tiles.

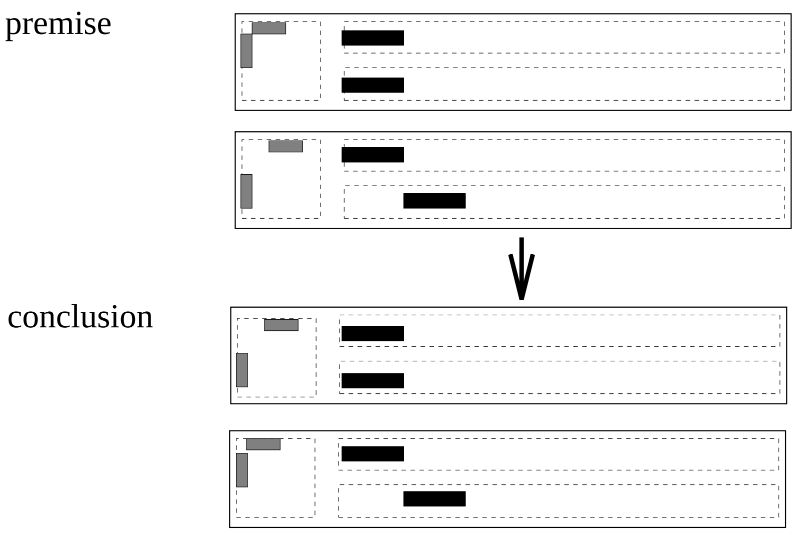


Figure 6.51: If the tile “1” is in the lower right corner and on the right side there is an empty cell, then move the tile “1” to the right.

Together, the question and the answer vector have the dimension 600 $(100 \cdot 3 \cdot 2)$.

After learning, a weight matrix of dimension 600^2 emerges (see fig. 6.52). The matrix is symmetric because for each association there exists an inverse association and vice versa. The matrix is low loaded because much of the space is not used for the representation. The weights are densely concentrated. The weight matrix is composed of only 6 elementary blocks and the sum values differ only slightly. There is a strong correlation of smaller pattern parts which declines as the pattern parts grow.

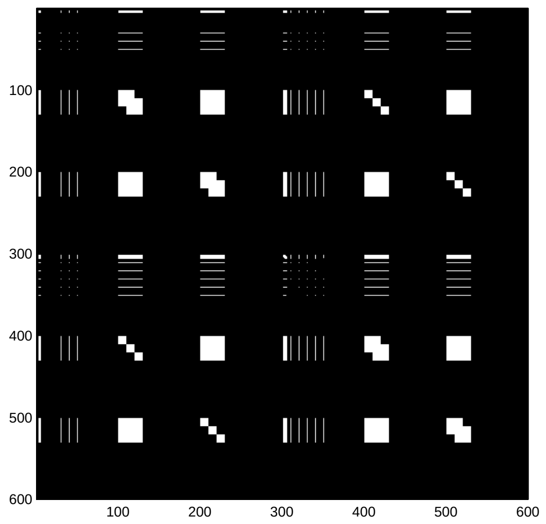


Figure 6.52: The weight matrix of the permutation associative memory for the task 8-puzzle. 15282 synapses are not zero, 4.2450% of the weight matrix.

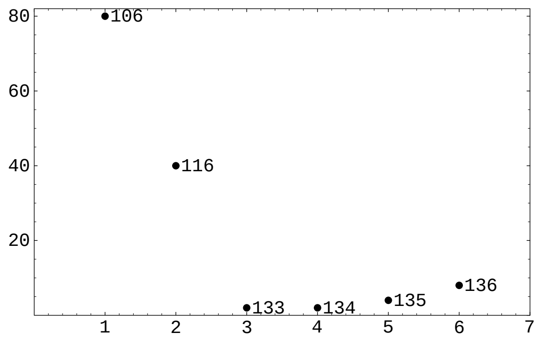


Figure 6.53: The sum of the synapses of a column. The x axis indicates the frequency of different sum values of columns. The written number corresponds to the sum values.

Superposition problem

Besides the learned, associations new associations emerge by superimposition. The new associations represent the not-allowed diagonal moves of tiles (see fig. 6.54). Their execution is prohibited by a constraint function. A pair of tiles which

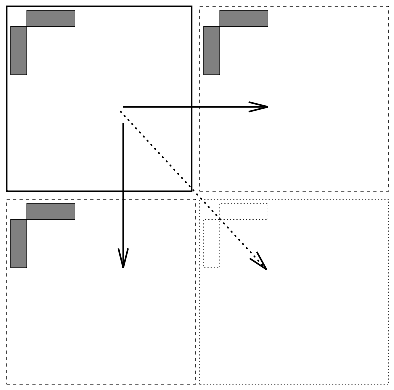


Figure 6.54: Besides the learned associations, new associations emerge by superimposition. The new associations represent the not-allowed diagonal moves of tiles.

lie in the diagonal have no overlap in the bars representing their positions[4] (see

[4]Associative fields two *and* three.

fig. 6.54). Two layers over nine cognitive entities are used. The hard threshold strategy is used with $threshold_{part} = 0.7$ and $threshold_{whole} = 0.87$. $1 + 9$ parts of associative memory were marked. The output of the computer simulation is shown during the attention stage with the input state the initial state I4 (see fig. 6.48 and 6.56).

```
Attention stage (threshold is 0.7):

Nr.1

No.1= -1    No.2= -1    No.3= -1    No.4= -1    No.5= -1    No.6= -1
No.7= -1    No.8= -1    No.9= 1

Nr.1

No.1= 1    No.2= 1    No.3= 1    No.4= 1    No.5= 1    No.6= 1
No.7= 1    No.8= 1    No.9= 0.921905
```

Associative memories were formed from three different marked $parts(i)_j$. Notation: i represents the position, j the number. The constraint combinations which are not valid are marked by `c`.Their qc_{whole} value is set to -1. The qc_{whole} values over the $threshold_{whole}$ are marked by `+`.Two possible answers were recognized.

```
Binding stage (threshold is 0.87):

9, 1,   has qc=-1 c
9, 2,   has qc=-1 c
9, 3,   has qc=0.365217
9, 4,   has qc=-1 c
9, 5,   has qc=-1 c
9, 6,   has qc=1 +
9, 7,   has qc=0.365217
9, 8,   has qc=1 +
```

Two possible moves were recognized.

Pattern heuristic

The $qc^2_{Ca}(b)$ function, heuristic function $h2$, is not used because in the 8-puzzle world no partial blindness (as in the block world) exists. The heuristic unmodified function $h1$ is used (see paragraph). The $h1$ function is equivalent to the city-block distance. The distance between a state and a desired state corresponds to the sum of distance by which the tiles are out of place. The closeness or similarity of a tile to the desired position of the tile is determined by the overlap of the two bar codes representing the tile (see fig. 6.55). The overlap corresponds to

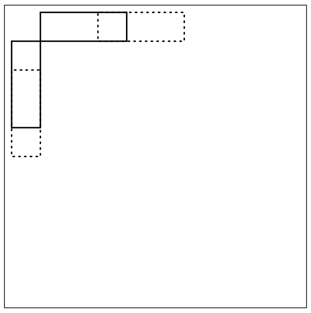


Figure 6.55: The tile "1" at the position of the tile "6" (shown by dotted bars). The value of the city-block distance is three. The hamming distance between the patterns representing the tile "1" and "6" is also three.

the distance by which the tile is out of place. The $h1$ heuristic emerged by a reasonable internal representation of the states in the 8-puzzle world. The initial states are represented in fig. 6.56. The desired state with the corresponding internal representation is shown in fig 6.50.

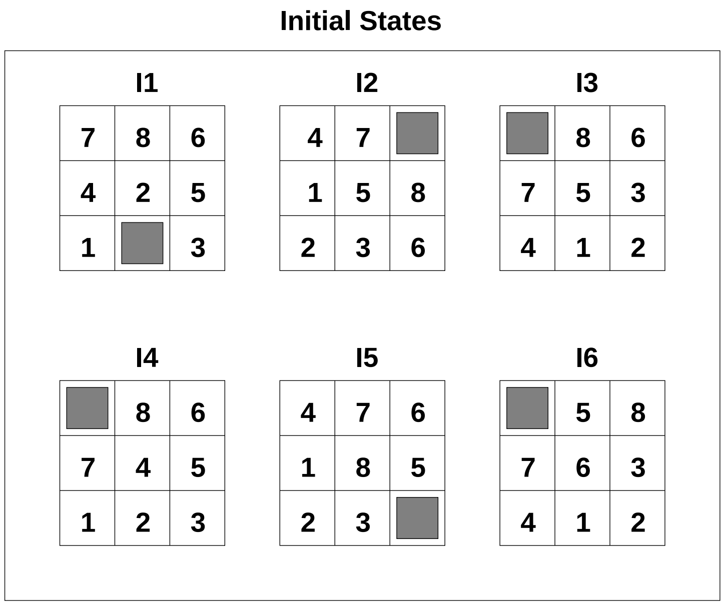


Figure 6.56: The initial states for the task 8-puzzle.

From the state I4 to the desired state the solution was found in 4 steps using the $h1$ heuristic (see fig. 6.48). The desired state was found in eight steps from the initial state I2 using the $h1$ heuristic (see fig.6.57).

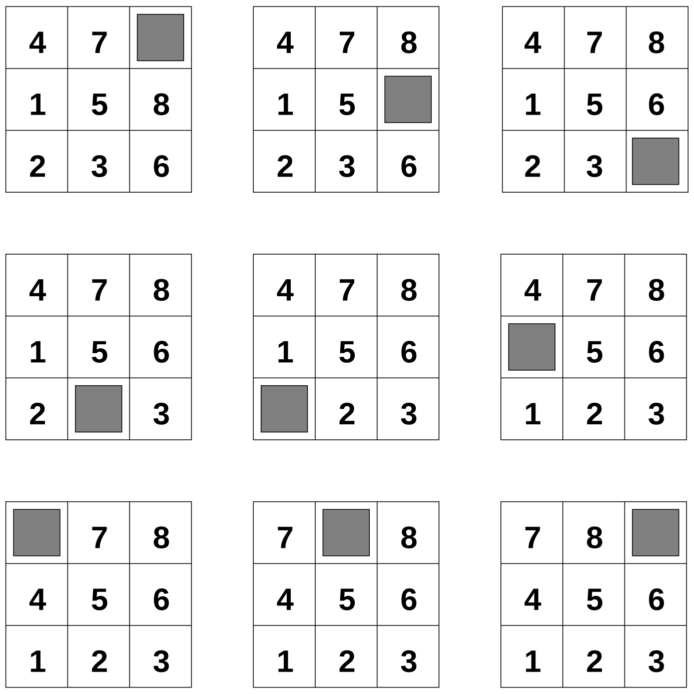


Figure 6.57: Planning 8-puzzle I2 task with h1 heuristic.

In fig. 6.58 we see the internal representation of the previous problems as used by the associative computer.

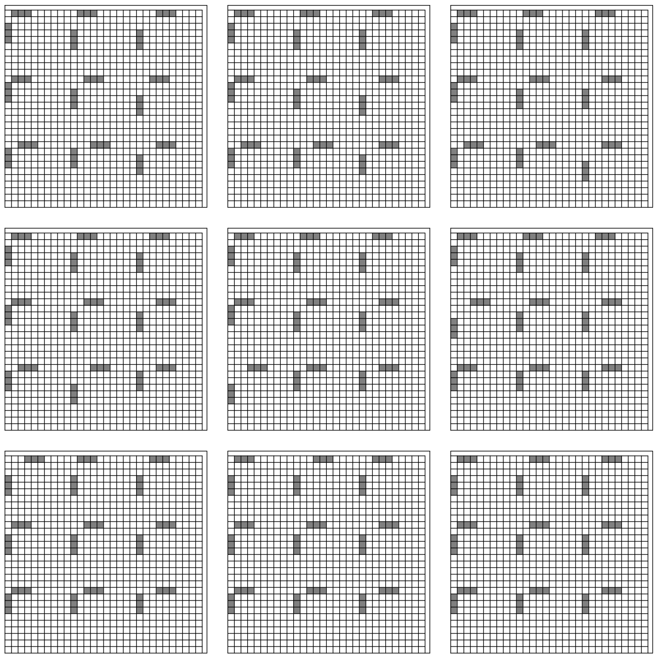


Figure 6.58: Planning 8-puzzle I2 task with h1 heuristic, internal representation.

6.2.3 Supervised learning

Six problems were solved with the aid of the pattern heuristic $h1$. These tasks were also learned by the prediction associative memory. They were recalled without errors.

task	h1	*all*
I1 → D	3	3
I2 → D	8	8
I3 → D	6	6
I4 → D	4	4
I5 → D	8	8
I6 → D	8	8

Table 6.15: 6 tasks for 8-puzzle.

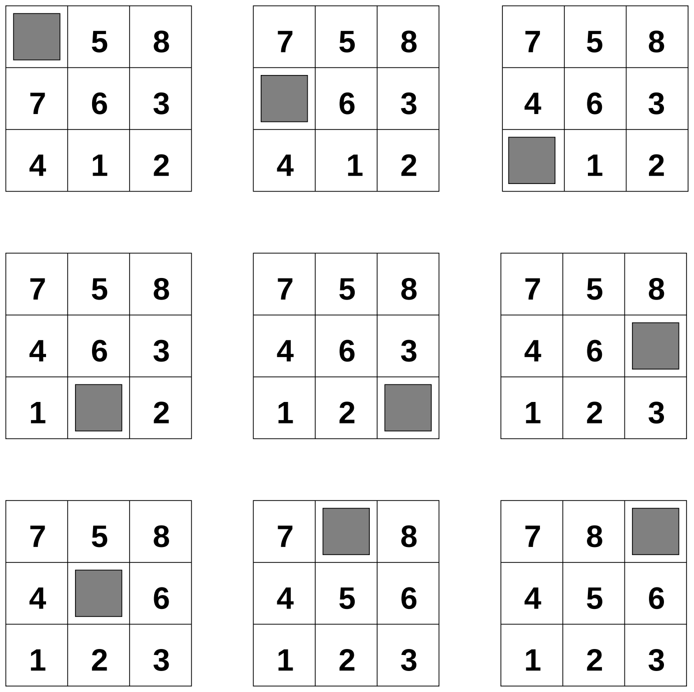


Figure 6.59: Planning 8-puzzle I6 task with h1 heuristic.

6.2.4 Generalization after supervised learning

An additional 28 initial states were used to test the generalization of the prediction associative memory trained by the six examples of problems (see fig. 6.60 and fig. 6.61). The computer resources were restricted to 30 steps. The performance of the prediction associative memory was nearly the same as that of the pattern heuristic $h1$. The "description" column in table 6.16 shows, for each of the not learnd initial states, its amount of difference from one of the learned example initial states. The number of *steps* here indicates the number of steps of the not learned initial state, counted from the initial state of the corresponding learned sequence. The description indicates the difference from the six learned problems characterized by the initial state. The value characterized by *steps* indicates the state, counted from the initial state of the learned sequence. The value *move* shows the number of moves of a tile from this state (see tab. 6.16). The prediction associative memory found solutions under the restricted computer resources, which the $h1$ heuristic could not find. For example, the solution to the problem I23 $\rightarrow$ D was found in 13 steps (see fig. 6.62).

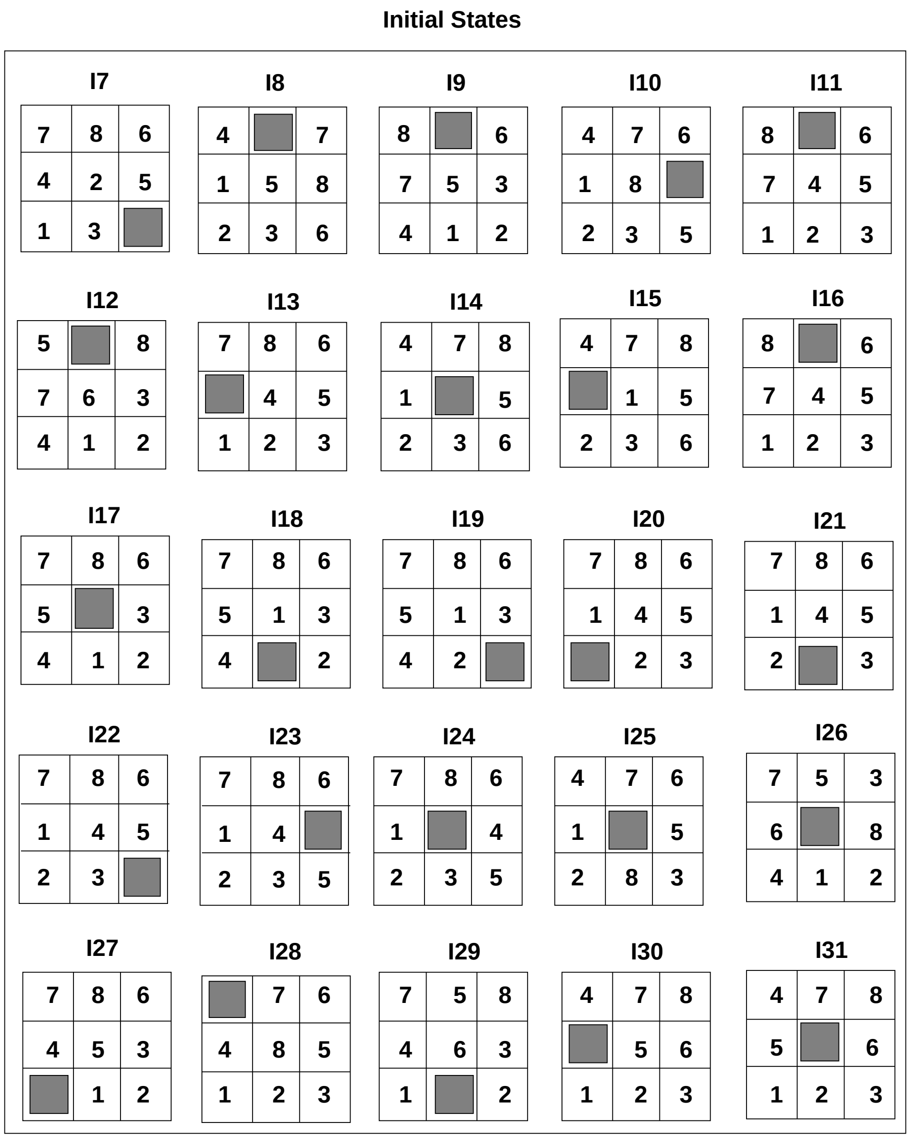


Figure 6.60: The not-learned initial states for the task 8-puzzle.

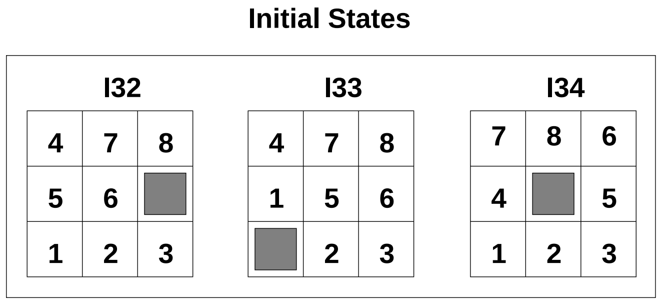


Figure 6.61: The not-learned initial states for the task 8-puzzle.

task	description	h1	all^{small}
I7 → D	from I1, one move	4	4
I8 → D	from I2, one move	9	9
I9 → D	from I3, one move	7	*none*
I10 → D	from I4, one move	none	none
I11 → D	from I5, one move	none	none
I12 → D	from I6, one move	none	none
I13 → D	one step from I1, one move	3	3
I14 → D	one step from I2, one move	8	8
I15 → D	one step from I1, two moves	9	9
I16 → D	one step from I1, three moves	5	*none*
I17 → D	one step from I3, one move	6	6
I18 → D	one step from I3, two moves	7	7
I19 → D	one step from I3, three moves	none	none
I20 → D	one step from I4, one move	4	4
I21 → D	one step from I4, two moves	5	11
I22 → D	one step from I4, three moves	6	12
I23 → D	one step from I4, four moves	**none**	13
I24 → D	one step from I4, five moves	none	none
I25 → D	one step from I5, one move	none	none
I26 → D	one step from I5, one move	none	none
I27 → D	two steps from I3	4	4
I28 → D	two steps from I4, two moves and four steps from I5	4	4
I29 → D	four steps from s6, one move	5	5
I30 → D	five steps from I2	**none**	3
I31 → D	five steps from I2, one move	**none**	4
I32 → D	five steps from I2, two moves	5	5
I33 → D	five steps from I2, two moves	4	4
I34 → D	five steps from I3 and two from I4, one move	2	2

Table 6.16: From 28 not learned states 10 (35.7%) where not recognized with h1 and 9 (32.4%) with all^{small}.

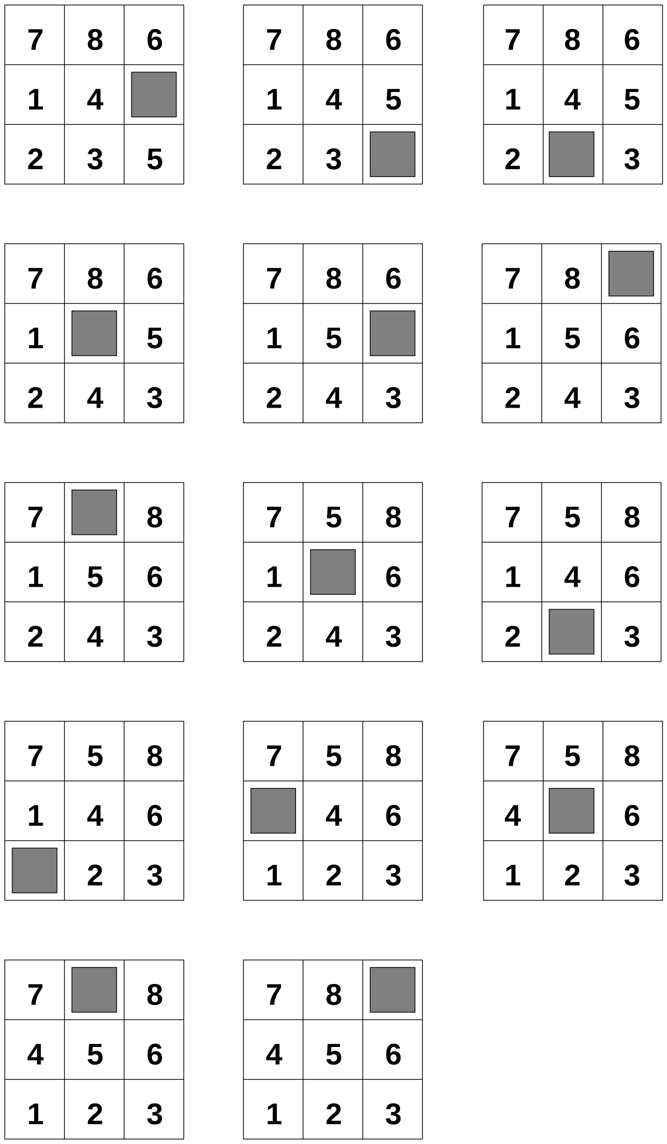


Figure 6.62: Planning 8-puzzle I23 after learning of the six example I1,..,I6.

6.2.5 Observations

- The building blocks of the permutation associative memory indicate a strong correlation of smaller pattern parts which declines as the pattern parts grow. The prediction associative memory is constructed conversely: there is a strong correlation of bigger pattern parts which decline as the pattern parts declines (see fig. 6.63).

- Representation of the tiles in the 8-puzzle world led to a pattern heuristic which is equivalent to the "city-block distance".

- Supervised learning with an observer who chose relevant examples led to results comparable to the pattern heuristic.

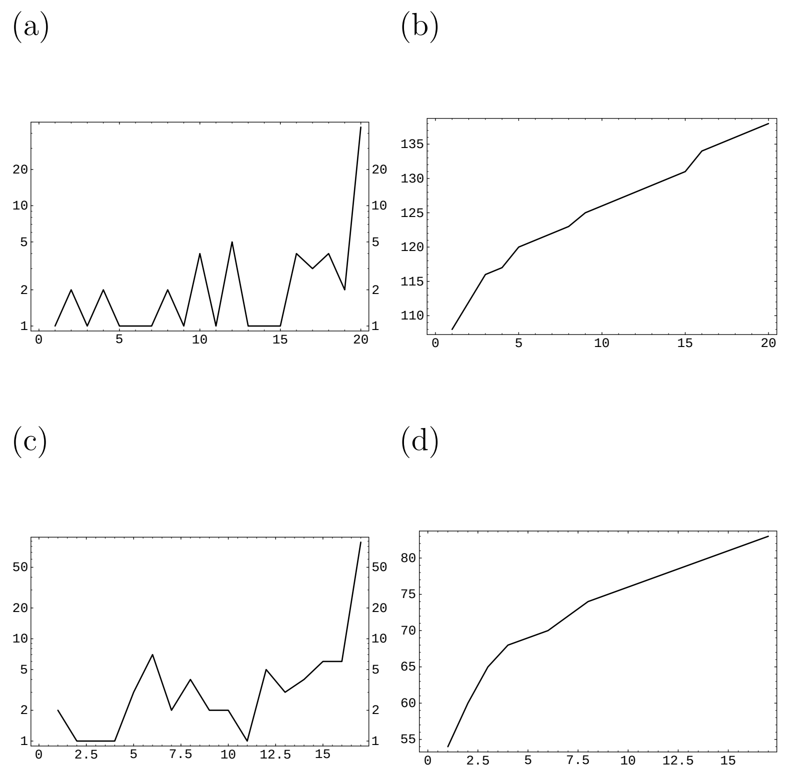


Figure 6.63: 0.006% synapses are not zero of associative memory of the task all^{small}. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows *(logarithmic plot)*. (d) The corresponding sum values of rows.

6.3 Robot

"Supposons par exemple que l'ont veuille représenter un monde dans lequelle un robot, Clotaire, est capable de se déplacer de pièce en pièce.." [43], page 163. A robot can move from one room to another and can move in four directions (north, south, east, and west). It can move into room only if a passage exists. The rooms with the passages are defined by a map. This map is defined by a sketch. The different rooms have different names (see fig. 6.64).

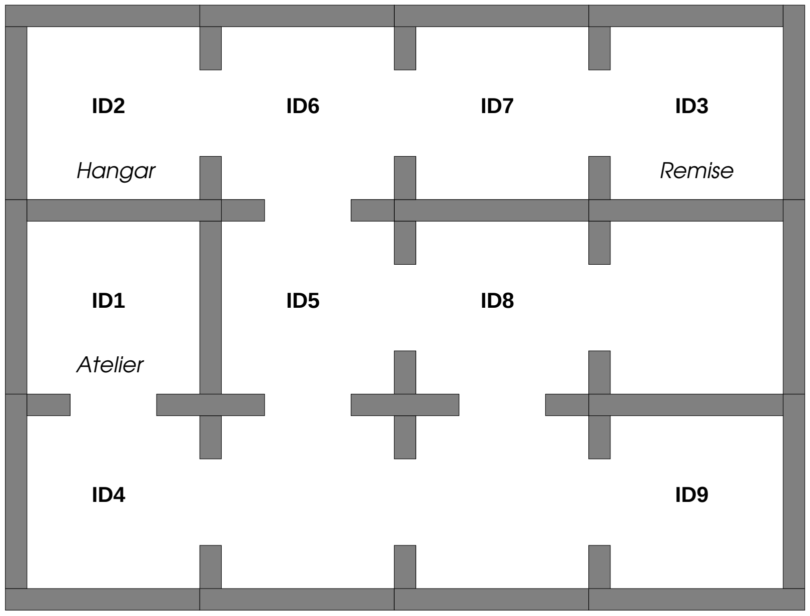


Figure 6.64: The initial and desired states for the task robot. The map is defined after example from [43].

6.3.1 Pattern representation

The robot is represented by one cognitive entity. It is specified by the first associative field. Its position is specified by bars in the associative fields two and three. The binary bars are of the size two. The maze is represented by 23 cognitive entities and it is decomposed into objects corresponding to walls, the intersections of walls, and passages. A object is specified by the first associative field, its position by the two remaining associative fields. The binary bars are of the size ten, due to the larger size of the objects. In total, a state is represented by 24 cognitive entities. In the following example the robot stands in the *Atelier*. The cognitive entity representation is shown below, followd by the pattern representation (see fig. 6.65).

```
No:1

0000000000
0000000000
0000000000
0000110000
0001111000
0001111000
0000110000
0000000000
0000000000
0000000000
```

```
0000000000000000000000110000000000000000000000000000000000000000000000000000000000000000000000000000

0011000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

No:2

0000000000
0000000000
0000000000
0000000000
1100000011
1100000011
0000000000
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

No:3

0000110000
0000110000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000110000
0000110000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:4

0000110000
0000110000
0000110000
0000110000
1111111111
1111111111
0000110000
0000110000
0000110000
0000110000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:5

0000110000
0000110000
0000110000
0000110000
0000110000
0000110000
0000110000
0000110000
```

```
0000110000
0000110000

0000000000000000000011111111110000000000000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:6

0000110000
0000110000
0000110000
0000110000
1111111111
1111111111
0000110000
0000110000
0000110000
0000110000

0000000000000000000000000000001111111111000000000000000000000000000000000000000000000000000000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

No:7

0000000000
0000000000
0000000000
0000000000
1111111111
1111111111
0000000000
0000000000
0000000000
0000000000

0000000000000000000000000000001111111111000000000000000000000000000000000000000000000000000000000000

1111111111000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000

.
.
.

No:21

0000110000
0000110000
0000110000
0000110000
1111111111
1111111111
0000110000
0000110000
0000110000
0000110000

0000000000000000000000000000001111111111000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000011111111110000000000000000000000000000000000000000

No:22

0000110000
```

```
0000110000
0000000000
0000000000
0000000000
0000000000
0000000000
0000000000
0000110000
0000110000

0000000000000000000000000000000000000000111111111100000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000011111111110000000000000000000000000000000000000000

No:23

0000000000
0000000000
0000000000
0000000000
1111111111
1111111111
0000000000
0000000000
0000000000
0000000000

0000000000111111111100000000000000000000000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000001111111111000000000000000000000000000000

No:24

0000000000
0000000000
0000000000
0000000000
1111111111
1111111111
0000000000
0000000000
0000000000
0000000000

0000000000000000000000000000001111111111000000000000000000000000000000000000000000000000000000000000

0000000000000000000000000000000000000000000000000000000000001111111111000000000000000000000000000000
```

6.3.2 Permutation associative memory

The associations describe the possible moves of the robot. The premise represents the position of the robot and the nearest passage way present, so that the robot can move through it. The conclusion describes the new position after the robot moved through the passage (see fig 6.66).

Four directions for every interior room are learned, two for the rooms at the four corners and three for the border rooms.

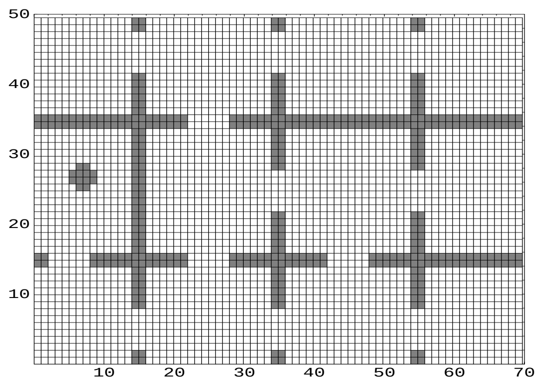


Figure 6.65: The world for the task robot. Robot stands in the *Atelier*.

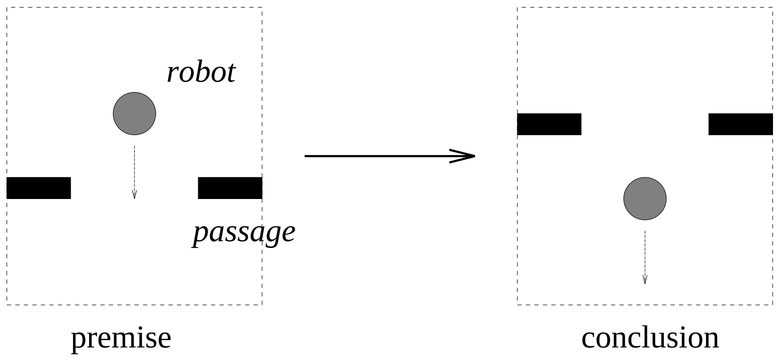


Figure 6.66: One association describes a possible move of the robot.

Superposition problem Besides the learned associations, new associations would emerge by superimposition. The symmetry is broken by representing the different states of the robot dependent on its move. Four states are defined: 00, 11, 10, 01. The following eight rules determine the new state after the robot moved:

- If the robot is in state 00 and it moves north or south, the new state is 11.
- If the robot is in state 00 and it moves west or east, the new state is 10.
- If the robot is in state 11 and it moves north or south, the new state is 00.
- If the robot is in state 11 and it moves west or east, the new state is 01.
- If the robot is in state 10 and it moves north or south, the new state is 01.
- If the robot is in state 10 and it moves west or east, the new state is 00.
- If the robot is in state 01 and it moves north or south, the new state is 10.

- If the robot is in state 01 and it moves west or east, the new state is 11.

A state corresponding to a two digit binary number is represented by the position of the robot in the room. This is possible because the represented rooms are big enough. The first "one" indicates a slight postponement of the movement of the robot in the vertical. The bar represented in the second associative field is moved by two components to the right if an initial "one" of the two digit number representing the state is present. Example of a bar:

```
1100000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

The bar is moved by two components if a "one" is present:

```
0011000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000000
```

The second "one" indicates if a slight postponement of the movement of the robot in the horizontal is realized. The bar represented in the third associative field is moved by two components to the right if a second "one" is present. The postponement of the position prohibits the emergence of superimposition.

Thirty-four associations are learned. Its premises and conditions are described by two cognitive entities. Together the question and the answer vectors have the dimension 600. The structure of the weight matrix represents the learned associations (see fig. 6.67). The positions of the robot are described by positions 101 to 200 and 301 to 400. In the corresponding diagonal sections the postponement of the positions which break the symmetry can be recognized. (Compare positions 401 to 500 and 501 to 600 which describe objects of the maze. The weight matrix is composed of 11 elementary blocks.) The correlation of smaller pattern parts prevails (see fig. 6.68).

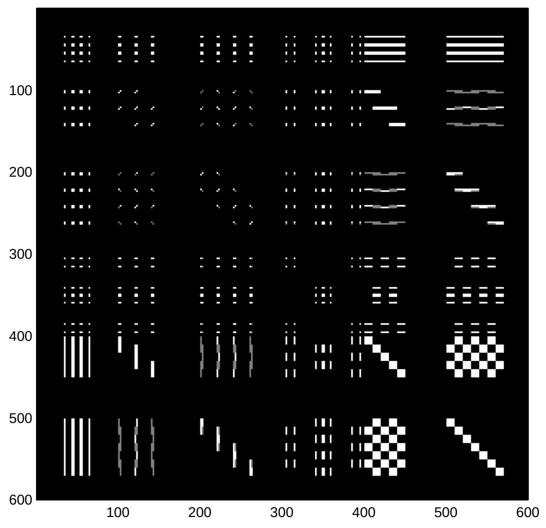


Figure 6.67: The weight matrix of the permutation associative memory for the task robot. 15632 synapses are not zero, 4.3422% of the weight matrix.

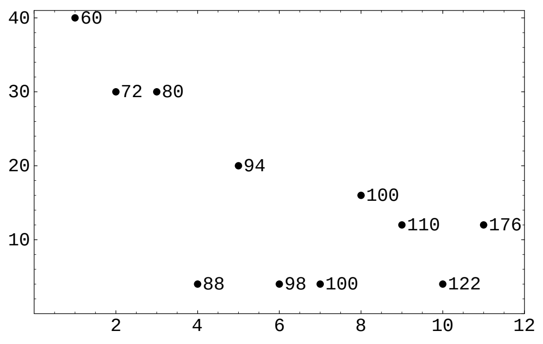


Figure 6.68: The sum of the synapses of a column.

Two layers of 24 cognitive entities were used. The $threshold_{whole}$ was increased to 0.93 to prohibit the evaluation of not-valid associations. In the following example the robot stands in the *Atelier* (see fig. 6.65):

```
Attention stage (threshold is 0.7):

Nr.1

No.1= 1    No.2= -1    No.3= -1    No.4= -1    No.5= -1    No.6= -1
No.7= -1    No.8= -1    No.9= -1    No.10= -1    No.11= -1    No.12= -1
No.13= -1    No.14= -1    No.15= -1    No.16= -1    No.17= -1    No.18= -1
No.19= -1    No.20= -1    No.21= -1    No.22= -1    No.23= -1    No.24= -1

Nr.1

No.1= -1    No.2= 1    No.3= 1    No.4= -1    No.5= -1    No.6= -1
No.7= -1    No.8= 1    No.9= 1    No.10= -1    No.11= 1    No.12= 1
No.13= -1    No.14= 1    No.15= 1    No.16= 1    No.17= -1    No.18= 1
No.19= -1    No.20= 1    No.21= -1    No.22= 1    No.23= -1    No.24= -1
```

First layer recognizes the robot, the second the passes.

```
Binding stage (threshold is 0.93):
```

```
1, 2,   has qc=1 +
1, 3,   has qc=0.896667
1, 8,   has qc=0.896667
1, 9,   has qc=0.471111
1, 11,   has qc=0.85
1, 12,   has qc=0.725714
1, 14,   has qc=0.85
1, 15,   has qc=0.471111
1, 16,   has qc=0.619677
1, 18,   has qc=0.471111
1, 20,   has qc=0.556667
1, 22,   has qc=0.471111
```

One possible answer was recognized. There is only one way out of the *Atelier*.

6.3.3 Unsupervised learning

The pattern heuristic $h1$ can not be used because no different overlaps exists between different states. The prediction associative memory was empty at the beginning of the unsupervised learning. Unsupervised learning of the six tasks brought an improvement (see fig. 6.70), however not a significant one (see table 6.17). This is due to the interference of the problem *Hangar* → *Remise* with other problems (see fig. 6.71). Context information led to the same results as without context.

task	t0	t1	t2-t10
Atelier → *Hangar*	7	7	7
Atelier → *Remise*	10	10	8
Hangar → *Atelier*	5	17	5
Hangar → *Remise*	33	33	33
Remise → *Atelier*	8	6	6
Remise → *Hangar*	3	3	3
mean	11	12.67	10.33
sdev	11.05	11.04	11.24
p	-	0.23	0.086

Table 6.17: Required steps.

task	t0-t10
Atelier → *Hangar*	7
Atelier → *Remise*	8
Hangar → *Atelier*	5
Hangar → *Remise*	3
Remise → *Atelier*	6
Remise → *Hangar*	3
mean	5.33
sdev	2.07

Table 6.18: Plan length.

task	t0	t1	t2-t10
Atelier → *Hangar*	0	0	0
Atelier → *Remise*	1	1	0
Hangar → *Atelier*	0	7	0
Hangar → *Remise*	15	15	15
Remise → *Atelier*	1	0	0
Remise → *Hangar*	0	0	0
mean	2.83	3.83	2.5
sdev	5.98	6.11	6.12

Table 6.19: Backtracking steps.

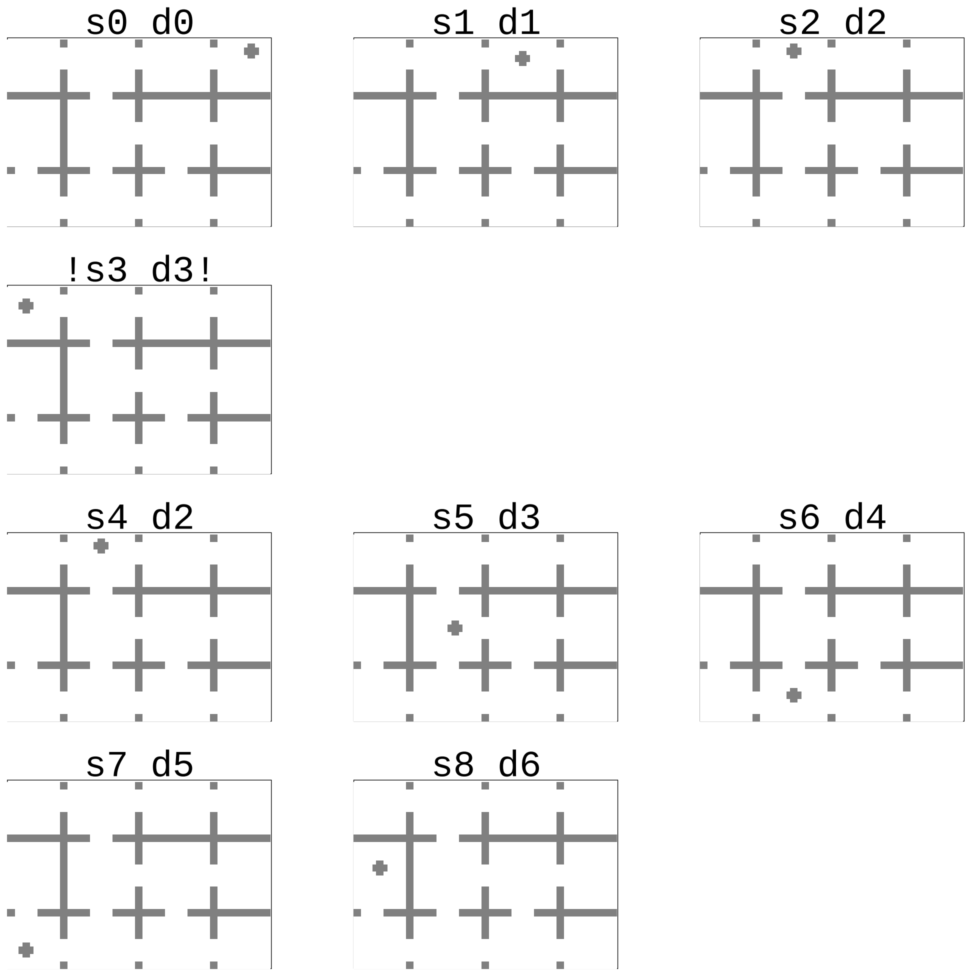


Figure 6.69: Planning of the robot task *Remise* → *Atelier*, blind search.

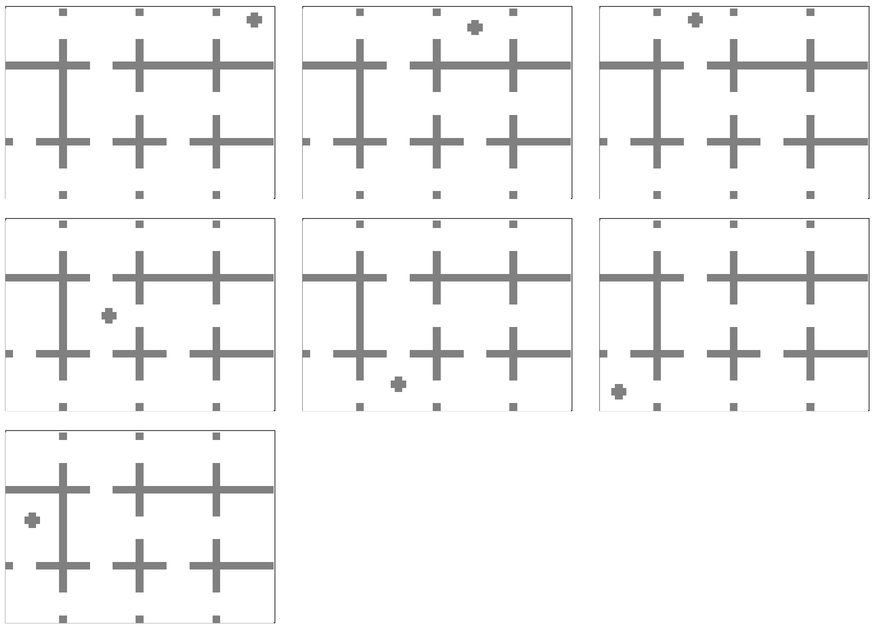


Figure 6.70: Planning of the robot task *Remise* → *Atelier* after unsupervised learning. The robot choses the shortest path.

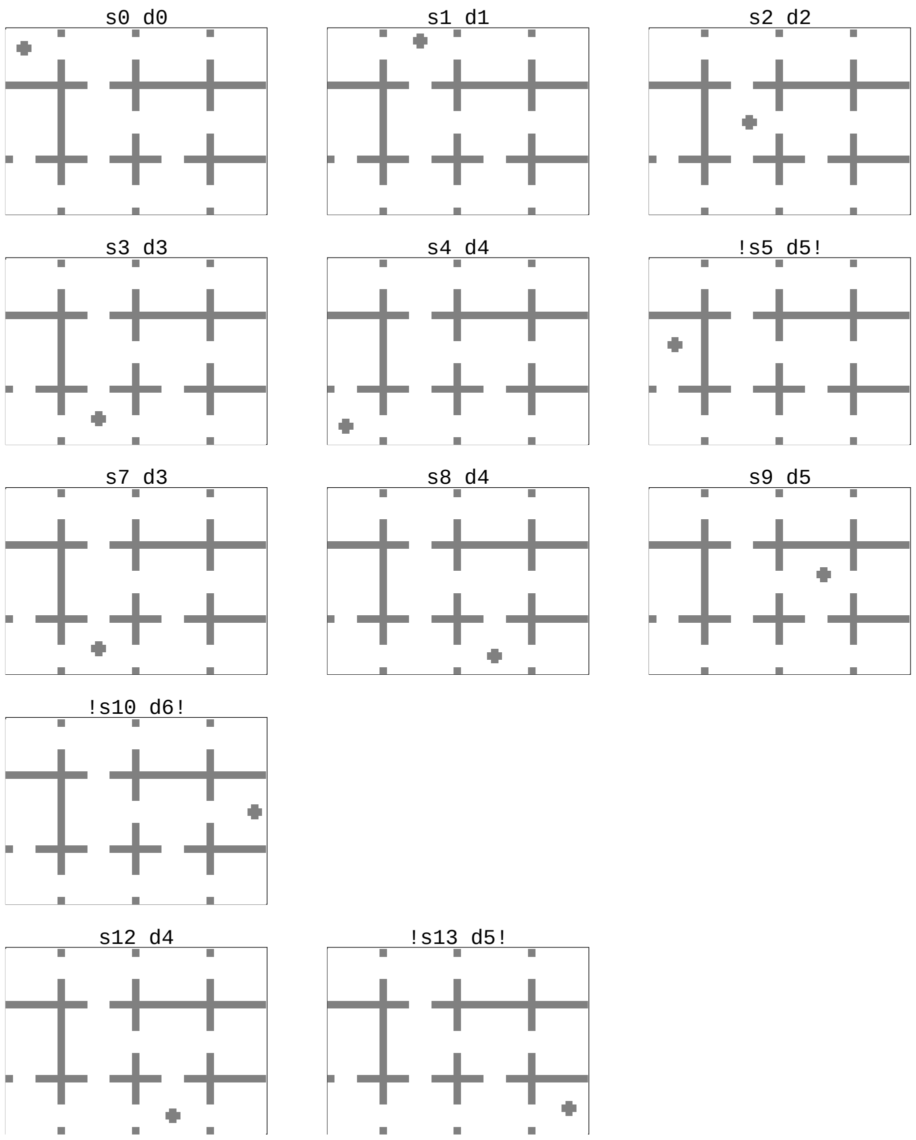


Figure 6.71: Planning of the robot task *Hangar* → *Remise* first part. Robot loses its way to the *Remise*.

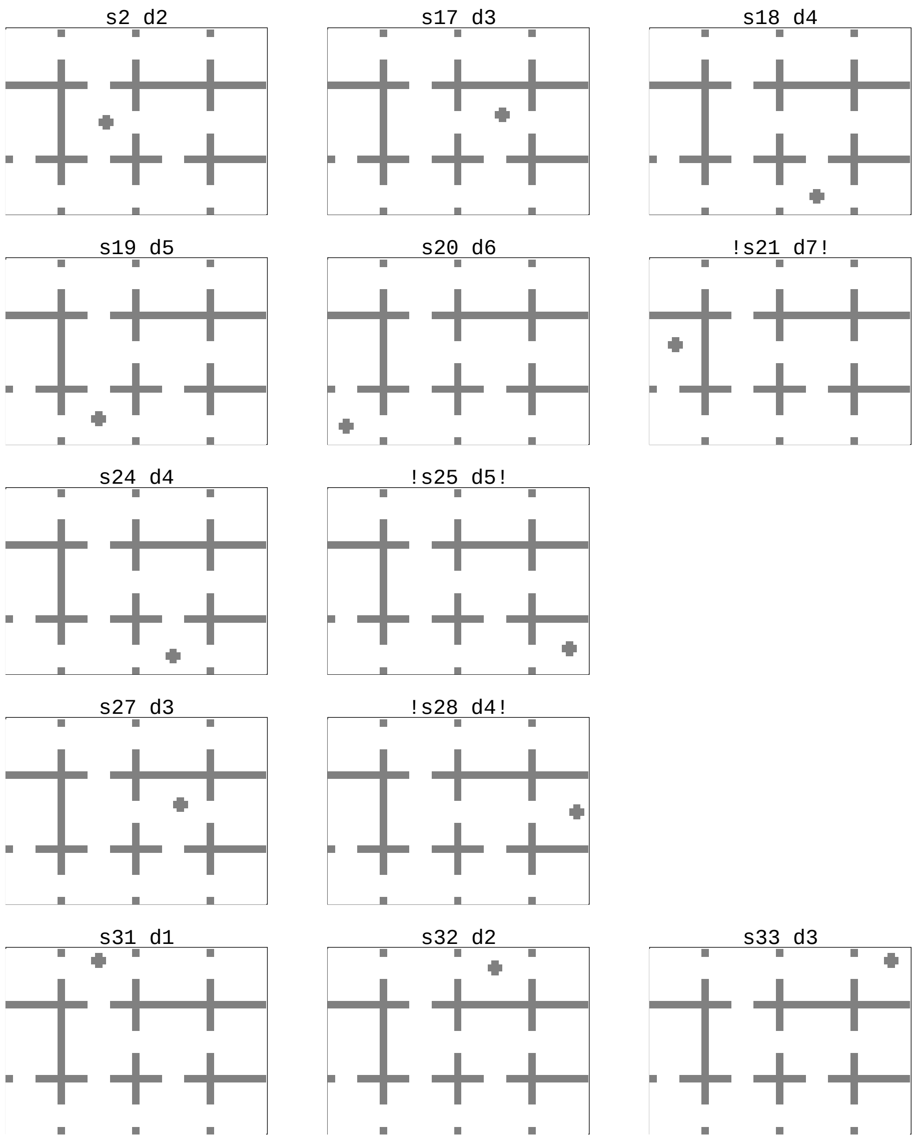


Figure 6.72: Planinng of the robot task *Hangar* → *Remise* final part.

6.3.4 Generalization after unsupervised learning

The 66 combinations of tasks in which the states ID4-ID9 were initial or desired states were used to test the generalization performance after the unsupervised learning. A significant improvement in the required steps was obtained. The improvement was obtained because often the robot recognized that he had already visited the rooms during previous explorations of the maze (see fig. 6.73).

	h0	us^{small}
mean	6.85	5.36
sdev	7.51	5.31
p	-	0.00017

Table 6.20: Required steps for all 66 not-seen combinations.

	h0	us^{small}
mean	3.18	3.21
sdev	1.65	1.66

Table 6.21: Required plan length for all 66 not-seen combinations.

	h0	us^{small}
mean	1.83	1.08
sdev	3.63	2.53
p	-	0.00012

Table 6.22: Required backtracking steps for all 66 not-seen combinations.

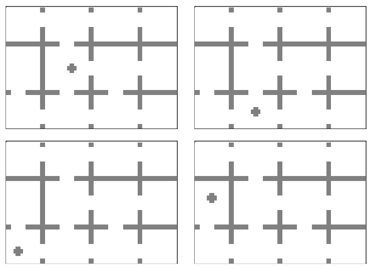


Figure 6.73: Planning of the robot task ID5 → *Atelier* after us^{small} learning.

6.3.5 Learning of all examples

All 72 combinations were learned. Already at the end of stage one, an equilibrium for needed steps and backtracking steps occurred. Plan length remained unchanged. The learning led to a significant improvement in the required steps and reduced backtracking steps. The results were better than after the learning of the six tasks. Context, however, led to worse results.

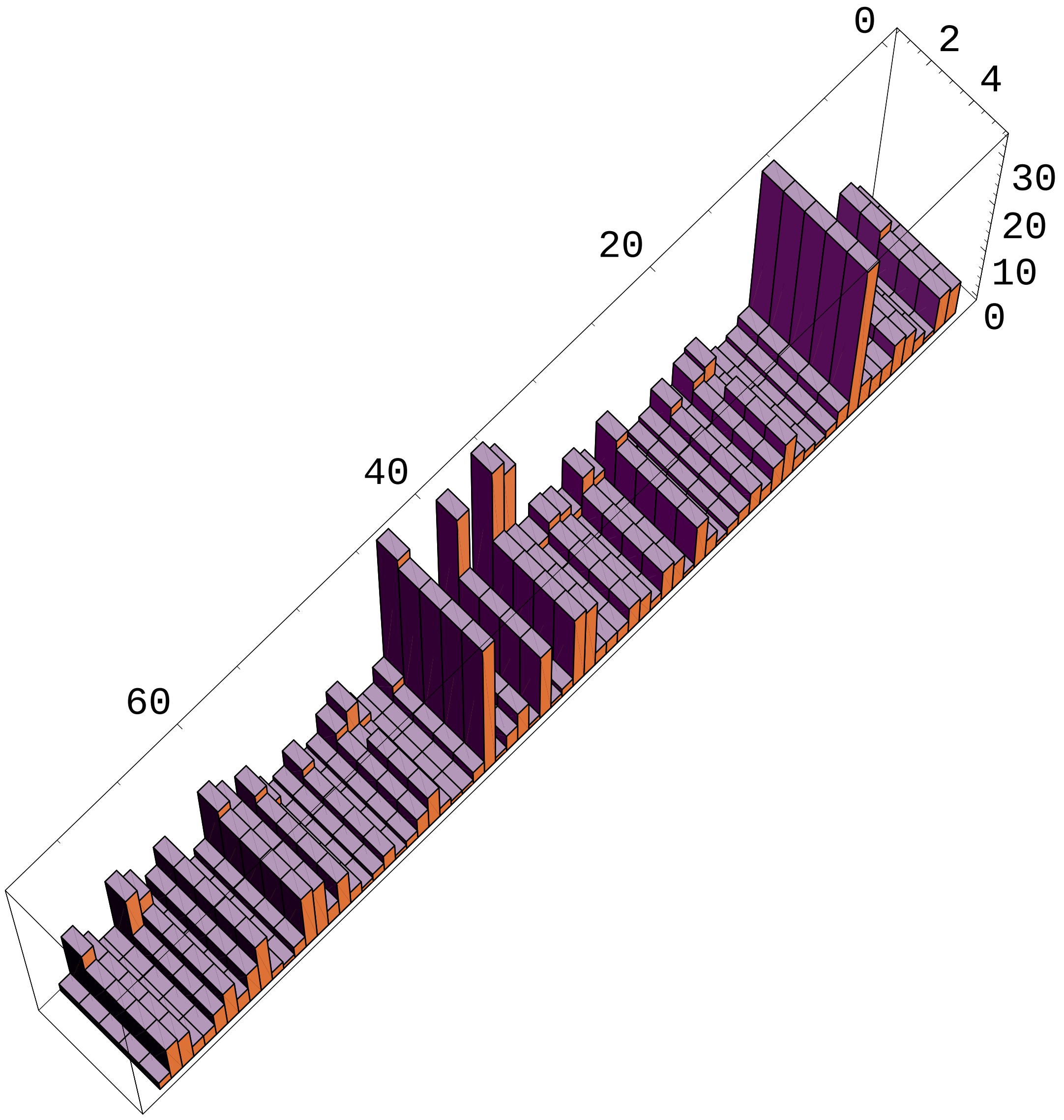


Figure 6.74: Required steps, big robot task during learning. Four main inteferences of problems with other problems can be seen.

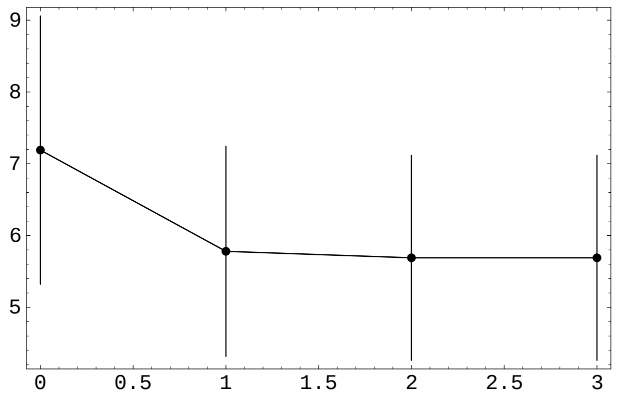


Figure 6.75: Mean steps for learning of all 72 combinations. *t0: mean=7.19, sdev=7.85; t2-t10: mean=5.69, sdev=6,* ***p=0.00008***

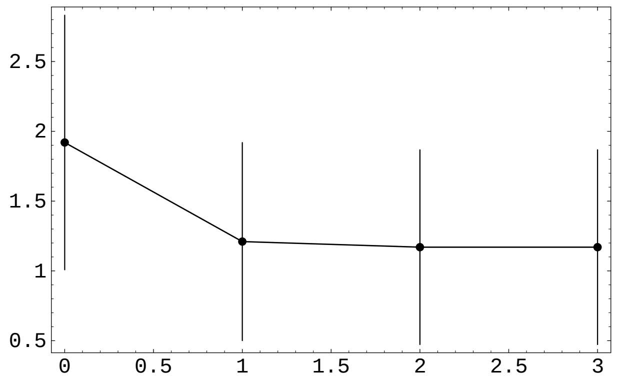


Figure 6.76: Mean backtracking steps for learning of all 72 combinations. *t0: mean=1.92, sdev=3.83; t2-t10: mean=1.17, sdev=2.93* ***p=0.00008***

	h0	us	c-us	us^{small}
mean	7.19	5.69	5.86	5.78
sdev	7.85	6	6.1	6.05
p	-	0.00008	0.001	0.0001

Table 6.23: Required steps for all 72 combinations.

	h0	us	c-us	us^{small}
mean	3.36	3.36	3.36	3.39
sdev	1.77	1.77	1.77	1.78

Table 6.24: Required plan length for all 72 combinations.

	h0	us	c-us	us^{small}
mean	1.92	1.17	1.25	1.94
sdev	3.83	2.93	2.96	2.94
p	-	0.00008	0.001	0.00004

Table 6.25: Required backtracking steps for all 72 combinations.

The improvement of learning all possible combinations as opposed to the learning of the six tasks resulted from the fact that the robot did not learn the path which included the room ID9 (see table 6.26).

task	us^{small}	us
ID9 $\rightarrow$ ID1	4	4
ID9 $\rightarrow$ ID2	11	9
ID9 $\rightarrow$ ID3	10	6
ID9 $\rightarrow$ ID4	5	3
ID9 $\rightarrow$ ID5	3	3
ID9 $\rightarrow$ ID6	4	8
ID9 $\rightarrow$ ID7	11	9
ID9 $\rightarrow$ ID8	2	2
mean	6.25	5.5
sdev	3.77	2.88

Table 6.26: The only differences in required steps between us^{small} and us. These differences include the state ID9 which was not learned by us^{small}.

The prediction associative memory is related to the prediction associative memory of the task 8-puzzle. It is constructed conversely to the permutation associative memory. There is a strong correlation of bigger pattern parts which declines as the pattern parts decline (see fig. 6.77).

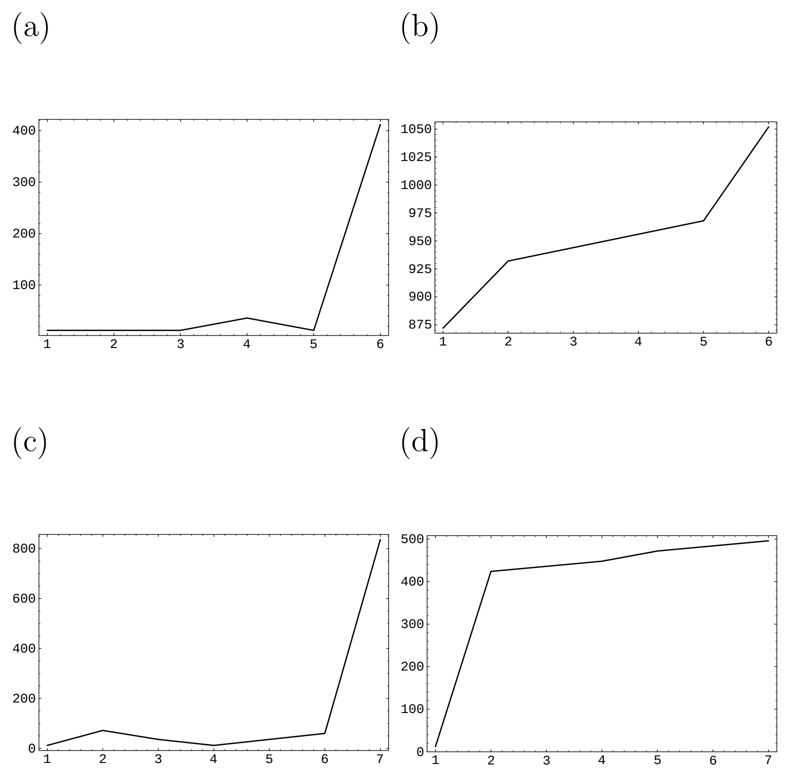


Figure 6.77: 0.2579% synapses are not zero of associative memory of the task us. (a) The frequency of different sum values of columns. (b) The corresponding sum values of columns. (c) The frequency of different sum values of rows. (d) The corresponding sum values of rows.

It is shown that despite the fact that a large quantity of information is present in which the important clues describing the problem are hidden (such as the position of the robot), the important information is extracted by the prediction associative memory and leads to a significant improvement in the problem solving behavior.

6.3.6 Noise

The handling of noise by the associative computer in the task robot in the maze was examined. Pixels of the associative field describing objects were either deleted or added. In the table 6.27 the required steps dependent on the kind of noise are shown. The $threshold_{whole}$ values were reduced to allow the treatment of noise. The correct valid associations are chosen with the aid of the prediction heuristic (us^{small}). The failure of the associative computer is indicated by *none*. The failure resulted either from not recognizing any associations or by producing mirage states (see fig. 6.78).

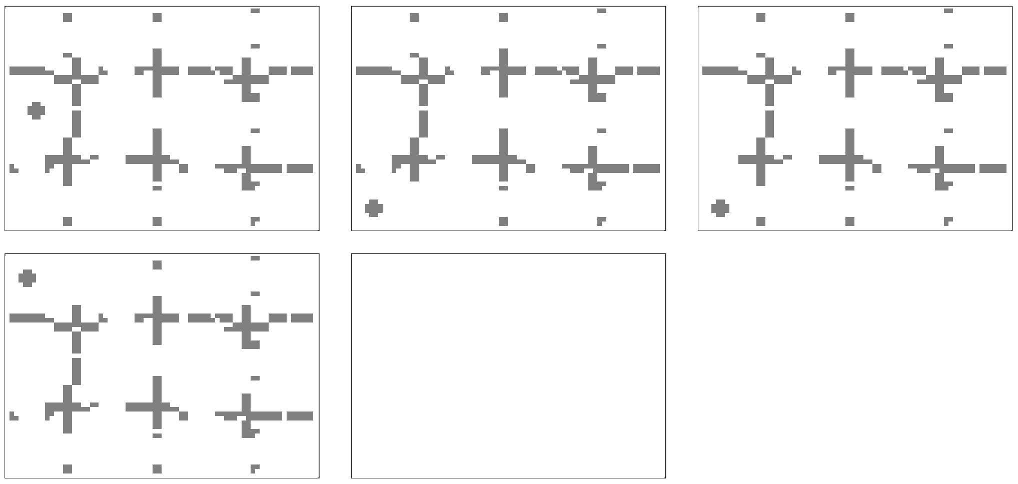


Figure 6.78: Failure of the task of robot *Atelier* → *Hangar*. The objects describing the maze are shifted and some pixels are deleted. The computations failed if objects were shifted.

task	0	-2	-2 & +1	-3	-3 & +1	-4	-4 & +1	-5
$treshold_{whole}$	0.95	0.93	0.93	0.88	0.88	0.85	0.85	0.85
Atelier → *Hangar*	7	7	5	7	7	none	none	none
Atelier → *Remise*	8	8	6	8	8	none	none	none
Hangar → *Atelier*	5	5	5	5	5	none	none	none
Hangar* → *Remise	3	3	3	3	3	*7 , 5*	none	none
Remise → *Atelier*	6	6	6	6	6	*8 , 6*	none	none
Remise → *Hangar*	3	3	3	3	3	*3*	*8 , 3*	none

Table 6.27: Required steps, plan length for different types of noise for the robot tasks after learning (us^{small}). The threshold value 0.85 is too low for correct computation. For values separated by a comma the first value shows required steps, the second shows the plan length.

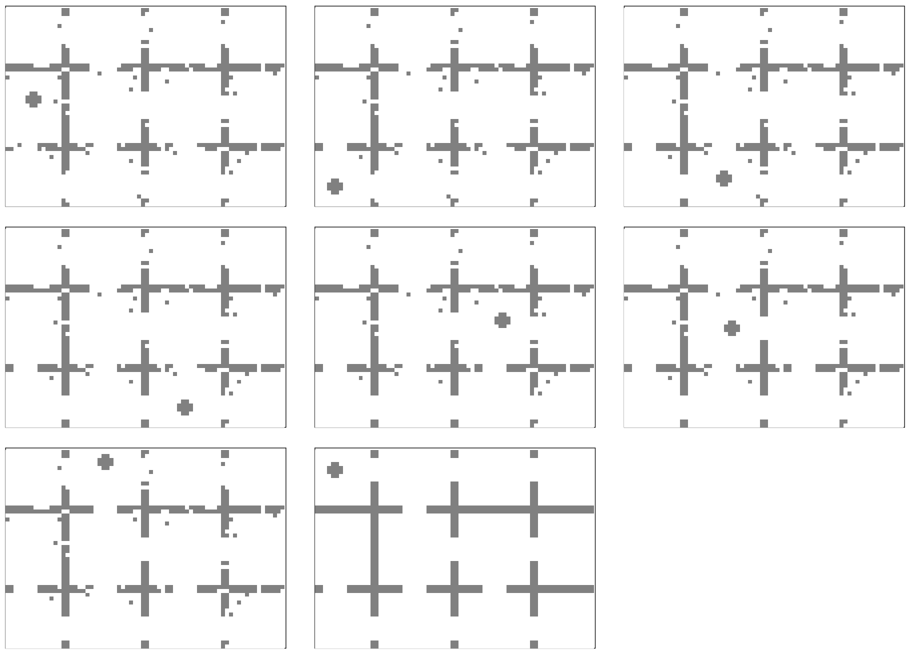


Figure 6.79: Planning of the robot *Atelier* → *Hangar* with noise after learning. The robot made a detour. This was not the case for the task *Atelier* ← *Hangar*.

6.4 Conclusion

A model which attempts to explain the process of human problem solving must not be constrained to only one area. It should have the same behavior in different domains. In this chapter it was shown that the associative computer is not constrained to a certain domain. Exemplary domains which are well known and extensively studied in Artificial Intelligence were examined. It was shown, how those domain can be represented with the aid of cognitive entities. It was also shown that the associative computer has the same behavior in those domains. It tolerates noise, learns, and uses heuristic information resulting from representation. The choice of the appropriate example as teaching material is very important. The prediction associative memory which was formed by learning the Sussman anomaly describes the problem of building the CBA tower statistically better than the pattern heuristic. One well-chosen example catches all the essential information about the nature of building the CBA tower. Unsupervised learning of many problems is difficult. This is because different states describing problems differ only slightly. Despite this fact, the prediction associative memory extracts the relevant knowledge which leads to a significant improvement in the problem solving behavior. The examined domains of problems indicate that the associative computer is qualified for usage in a robot, as robots must solve analogical problems which are presented.

6.4.1 Associative memories

The weight matrixes of the permutation associative memories were ordered and built by not many elementary blocks. The weights were not equally distributed over the whole matrix.

task	%	#*blocks*
geometric blocks	5.4874	10
tower	9.5738	12
ABC blocks	5.2807	7
9 blocks	11.0849	11
puzzle	4.2450	6
robot	4.3422	11

Table 6.28: Structure of the permutation associative memories. The percentage of synapses of the weight matrix which are not zero and the number of elementary blocks are shown.

Prediction associative memories were formed by associations which were represented by patterns. The weight matrixes were very low loaded. This was the case due to very high dimension which resulted from storage of whole patterns.

task	*y dim*	*x dim*
geometric blocks	19970	9970
tower	19990	9990
ABC blocks	19970	9970
9 blocks	19970	9970
puzzle	19521	9521
robot	19956	9956

Table 6.29: Dimensions for the prediction associative memory. For context 10000 is added to the *y dimension.*

Pattern representation of pictures also led to ordered weight matrixes as seen by the building blocks.

task	*%*	*#blocks*
tower	0.4962	30,31
abc blocks	0.0746	21,21
9 blocks	0.3695	31,37
puzzel	0.006	20,17
robot	0.2579	6,7

Table 6.30: Structure of prediction associative memories. The percentage of synapses of the weight matrix which are not zero and the number of elementary blocks are shown.

Both results indicate that the vectors corresponding to the visual categorical based representation in the problem solving domain are strongly correlated [5]. There is only a minimal difference if the vectors result from cognitive entity representation or pattern representation. This is because during problem solving the world changes only minimally. The resulting vectors are not sparse and the number of 1s is not as likely to be represented in coordinates of those vectors. The storage capacity is much lower, but despite this fact associative properties were preserved, as was shown by the prediction heuristic and experiments with noise.

A question remains open: How is the equilibrium of the prediction associative memory connected with the behavior of the associative computer?

weak hypothese 1 *During a successful unsupervised learning, the equilibrium of the prediction associative memory is reached, mostly after the stabilization in the behavior of the associative computer.*

[5]Task dependent coding: Representation of features which are relevant to a special problem leads to better results.

It was shown how superimposition can be treated:

- by constraints.
- by perturbation of the patterns which were learned.

6.4.2 Pattern heuristic

The division of the behavior of the model into basic behavior and additional property behavior resulting from the pattern and prediction heuristic corresponds to the Michalskis two tier philosophy of concept meaning [120, 121, 89].

It was shown that visual-categorical based representation led to a pattern heuristic which increased the performance of problem solving significantly.

The basic behavior of associative computer models corresponds to the behavior of a symbolic production system which performs a depth-first search strategy. Depth-first search strategy is a blind search strategy *(h0)*. It was shown by experiments that an associative computer using the pattern heuristic *(h2)* needs on average significantly fiewer steps than one using a blind search strategy *(h0)* (see table 6.31). Because the pattern heuristic compares the similarity of pictures representing states "Thesis 1" follows.

task	examples	h0 steps	h2
geometric blocks	30	14.4	27.29%
ABC blocks	42	56.29	58.47%

Table 6.31: For a certain task for which a number of examples was presented, shown in column *task* and *examples*; the blind search strategy *h0* needed on average the number of steps shown in column *h0 steps*; the pattern heuristic *h2* brought a significant improvement in % which is shown in column *h2*.

Thesis 1 *Shape similarity between representation of states through visual categories is significantly similar to the distance in the problem space.*

6.4.3 Prediction heuristic

An associative computer which uses the prediction heuristic needs on average significantly viewer steps, then when using a blind search strategy *(h0)* (see table 6.21). Because the prediction heuristic was formed during learning and is dependent on the knowledge which is represented by pictures "Thesis 2" follows.

Unsupervised learning *(us)* led to slightly worse results then the pattern heuristic. Supervised learning with an observer *(sup)* brought the best results.

The observer chose relevant examples which described the problems as well as possible.

task	examples	h0 steps	h2	us (c-us)	sup
geometric blocks	9	14.89	13.43%	13.43% (20.89%)	47.75%
ABC blocks	34	62	57.68%	40.52% (46.48%)	81.21%
robot	72	7.19	none	20.86% (18.4%)	none

Table 6.32: For a certain task for which a number of examples was presented, shown in column *task* and *examples*; the blind search strategy *h0* needed on average the number of steps shown in column *h0 steps*; the pattern heuristic *h2* brought a significant improvement in % which is shown in column *h2*. The prediction heuristic which was formed by unsupervised learning without *(us)* and with context *(c-us)* brought a significant improvement in % which is shown in column *us (c-us)*; the prediction heuristic which was formed by supervised learning *(sup)* brought a significant improvement in % which is shown in column *sup*.

Thesis 2 *Representation of states through visual categories enables the access of knowledge which was formed by learned experience during problem solving.*

Chapter 7

Comparison to Related Works

The neural networks which model human problem solving are characterized by different techniques. Depending on the technique, different abilities emerge, such as the ability to deal with information that is fuzzy, probabilistic and noisy. Other abilities include the ability of learning and the ability to work in parallel. The presence of all these abilities is desirable.

7.1 Techniques

7.1.1 Representation

Localistic representation

In localistic representation each concept is represented by a unit. This representation of knowledge by units is also called "grandmother coding". Because each unit represents a concept there is also a unit which represents the concept's "grandmother" [122, 32, 7, 207, 197, 59]. Localistic representation is widely used in categorization systems. A direct map of symbolic structures onto a neural network structure is possible with each symbol represented by a unit. The connections between units directly reflect the linkage between symbolic structures. A massively parallel system results, but it does not have, however, generalization properties.

Distributed representation

In distributed representation concepts are represented as distributed patterns [143, 116, 122, 7]. Distributed representation enables vector completion [143, 32]. The vectors may represent pictures or features.

Aleksander and Morton suggest the usage of icons for the representation of states [4]. Icons are pictures which cannot be broken into meaningful pieces and can be processed by a neural state machine.

Vectors composed of features were used by Steinbuch [194] for weather prediction and medical diagnosis with an associative memory. Since then, feature representation has been widely used. A "present" feature is indicated by a "one" at a certain position [114].

In "analogical representation" the features are represented in maps, which can be usually thought as simulations of important aspects of the environment of a robot [137].

Modular distributed representation

Localistic and distributed representations are present in real neural nets [68]. Smolensky suggested that both representations should be present in neural networks which model cognitive processes.

Another approach was suggested by James [78] and substantiated by Anderson [7]. Vectors are broken into meaningful pieces. These pieces are themselves vectors which involve operations and are therefore called by Anderson "cognitive entities"[7]. Smolsky designates them "knowledge atoms" [186].

Skapura used three units to represent three states for each symptom (Present, Absent, Unknown) [182]. Only one unit per symptom was activated. Units associated with symptoms that were unknown or irrelevant were set to the "unknown" state. After pattern completion by the neural network, the learned pattern which best matched some partial input pattern was presented. Complete symptom patterns could in this way be reconstructed.

7.1.2 Binding problem

The "binding problem" determines how to connect together all physically separated fragments of a complex object so that they can be processed as a whole by a neural network [122, 95, 219, 74]. For example, a red block is obviously a different object then a blue block. The fragments in this example are the form and the color.

Markers Discrete symbols called markers are propagated throughout the network [29, 162, 138, 105]. For each complex object a different marker exists which represents the corresponding group of fragments. Markers are identified with transmitters [19] or signals [104].

Synchronicity Several researchers suggested the hypothesis that synchronicity acts as a binding mechanism in real neural networks [39, 51, 181, 180]. When two groups of neurons oscillate in synchrony, a bond between them is formed. Ajjanagadee and Shastri [174] suggested the use of synchronicity instead of markers. Different fragments are bound together by simultaneous activity. The number of simultaneous bindings is limited by the available phase.

Tensor product Tensor product bindings are represented in a matrix with objects in one dimension, and their properties in another, with entries where object and properties meet [35, 187]. Several bindings can be superimposed by summing their tensor representation (see fig 7.1).

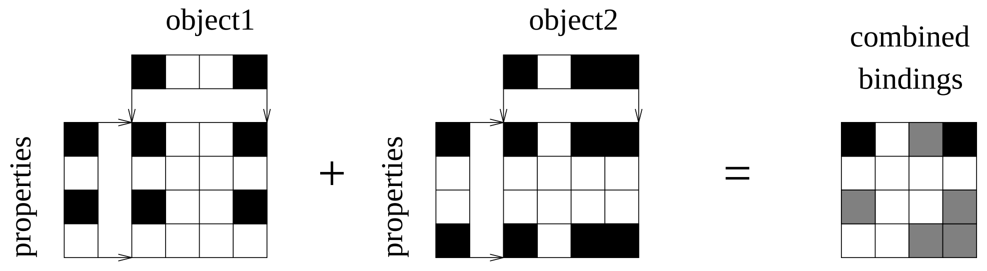


Figure 7.1: Tensor product bindings.

Concatenation Vectors representing different fragments are concatenated to a vector representing the object. This kind of binding was realized by the cognitive entity representation described in this work. This idea is motivated by the biological hypothesis that in convergence zones which consist of ensembles of neurons, the knowledge is bound together [210, 34]. This corresponds with the suggestion that different objects are represented in different sub-networks [145, 158].

Markers and synchronicity are mainly used in models which use localistic representation. Tensor product is used in models which use distributed representation. Models which use modular distributed representation use concatenation.

7.1.3 Learning search control knowledge

There are different kinds of learning laws which can help a problem solver to learn from past experience [21, 127]. “Chunking” and “learning by explanation” are learning laws which were symbolically described and implemented, and integrated into a symbolic problem solver. Folding architectures, analogy learning and associative prediction are neural methods which form, together with a symbolic problem solver, a hybrid architecture.

Chunking

Chunking is a learning method which is used in the SOAR system [99, 133]. It operates by summarizing the information examined while processing a subgoal. If a state causes an impasse a new rule is learned which avoids it. An impasse is present, for example, when no valid succeeding state exists. By the generalization of new rules future impasses are avoided.

Learning by explanation

Explanation-based learning can produce control knowledge by analyzing the trace of problem solving examples. The system explains why the choices were made and the explanation identifies the relevant features of the example [128, 126]. A strong body of domain knowledge should be present as it is useful in explaining both the problem and the generalization of that explanation. After learning, a description is present which is a generalization of the example and which helps to control the search of other examples.

Folding architecture

States described by logic formulas can be represented by trees. The folding architecture network [93, 58, 92] is closely related to the recurrent neural network and RAAM [154, 191]. It allows the representation of labeled trees. Each label is represented as a random real vector and all vectors are disjunct and of a fixed length. Similarity between tress can be defined in this manner. A heuristic evaluation function can be learned which computes numeric ratings for states. All states lying on a solution path are used as examples with positive target ratings, while all states lying close together in the search tree are used as negative ratings. The heuristic evaluation function was used by an automated deduction system SETHEO [129] on the "world problems in group theory". Of 317 different world problems in the group theory, 298 were solvable by SETHEO. The length of the proofs varied between one and eleven inference steps. Several neural folding architectures were trained on these examples. After the training, 16 of the remaining 19 proofs could be solved [58].

Analogy learning

A recurrent neural network architecture learns analogies by examples [20]. Examples consist of analogy problems with solutions. Geometric spatial analogies between states composed of different objects at different positions and different colors are learned. The states are represented by the tensor product between two vectors. The first vector describes the objects by feature coding, the second their position. The analogy between two scenes corresponds to the learned similarity between them. Experiments were also performed on letter part analogies and family tree relations [71].

Associative Prediction

Günther Palm suggested the usage of the associative memory to associate a situation with a list of moves, including in the association of the relative value of those moves [143]. This idea was realized in the domain of game playing. A heuristic for checkers and chess was learned with the aid of the associative memory [112, 9].

In checkers, a game position represented the question. The succeeding position and its value computed by the MINIMAX algorithms was the answer vector. The associative memory learned game states and the succeeding game states (with its relative values) by diverse example games. After training, the associative memory was used to retrieve the succeeding learned positions with their values during a new game. A retrieved succeeding position reduced the needed search so that an improvement in the game playing was observed. The game's positions were coded by feature vectors and a figure was coded by a "one" at a certain position. A vector was composed of vectors who indicated, for each position, the figure which occupied it. The numeric values resulting from the MINIMAX algorithm were coded by a "one" at certain positions. The analogic method was used, on the other hand, for chess play [9]. It led to the reduction of the size of the search trees. The prediction associative memory of the associative computer also codes succeeding states, however, it does not include any evaluation values.

Also Nils Nilsson [137] suggests the usage of a neural network to predict the value of a state represented by a feature vector. *After training, the prediction network can be used to compute the feature vectors that would result from various actions. These in turn could be used as new inputs to the network to predict the feature vector two steps ahead, and so on.* From [137], page 123.

7.2 Neural Systems

The main focus of statistical methods is on the mathematical description of the learning methods, and not on their realization[1]. The perspective under which they are discussed here is on their lack of defined interior representation. Other approaches orient themselves on methods which were studied by artificial intelligence and use defined interior representation.

7.2.1 Statistical methods

Markov Models

The Markov process generates a series of states. If each state depends only on the preceding state, it is called the "first order" Markov process [113]. Both, the choice made to move from one state to another depending on the state transitions, and these transitions themselves may be determined by training. The associative prediction heuristic corresponds to a first order Markov process. A state is only dependent on the proceeding state[2]. It would be useful to have a description of a state which also captures a notation of the status of the problem solver [13]. The status could be dependent on the executed steps, or the changing environment if

[1]On example of which is by recurrent networks.

[2]The desired state is included in the state description.

the problem solver is integrated in a robot. Different state transitions could be implemented depending on the status. A description of the model is in this case unknown and the model must be approximated by learned examples. Because the model is not known, it is called the "hidden" Markov model [79]. A powerful and provably convergent training algorithm exists for hidden Markov models. However, the model has not yet been used in the problem solver domain. It has been, however, successfully used in speech recognition, despite the fact that speech signals can not be approximated by a first order Markov process [79].

Reinforcement Learning

The consequence of actions may not be known beforehand. The value of taking the action is not known, because the rewards are delayed. For example, for a robot in a maze, the value of a taken action is only known after the robot reaches the desired room. So the value of an action must be discovered by experiments. The mathematics to handle delayed rewards is termed "reinforcement learning" [13, 203]. It is possible to make local improvements because the best local incrementally changing improvements in action are also the right thing to do globally.

Q-Learning is an example of reinforcement learning which is mainly used in Stochastic domains. $V_f(x)$ is the expected reward from the policy f to take action in state x. The policy is simply the action taken in state x. Let $Q(x, u)$ be the action value function which gives the expected return value of starting in state x and taking action u, and the following policy f thereafter. The $V_f(x)$ function is related to the Q-function by

$$V_f(x) = \max_u [Q(x, u)].$$

The Q-function can be calculated by the Q-learning algorithm. A system can also try to make consistency predictions. In this case, the learning is called the "temporal-difference" learning.

Q-learning was used to learn to pick up a block of a certain color [13]. The task is for the robot arm to move other blocks away from the block of the specified color so that it can be picked up. Deictic frame representation was used for this task instead of pattern representation. Q-learning was widely used in learning complex kinematics [87, 216]. Temporal-difference learning [202] was used to learn to play backgammon [13].

7.2.2 Partially recurrent networks

The connections of neural systems are mainly feed-forward but also include carefully chosen feedback connections. This recurrence lets the network "remember" cues from the past and there is a special set of context units that receive feedback signals. Because of these context units, the state of the network depends on the previous states as well as on the current input. Different architectures for implementation have been suggested, all of which produce similar behavior. Elman suggested that the context units are fed from the hidden layer[3] (see fig 7.2) [40]. Jordan suggested an architecture in which the context units are fed from the output layer and also from themselves (see fig. 7.2) [80]. Hammer showed that that Elman networks are universal approximators for sequences [63]. It was also shown that slightly modified Elman networks posses Turing power [176].

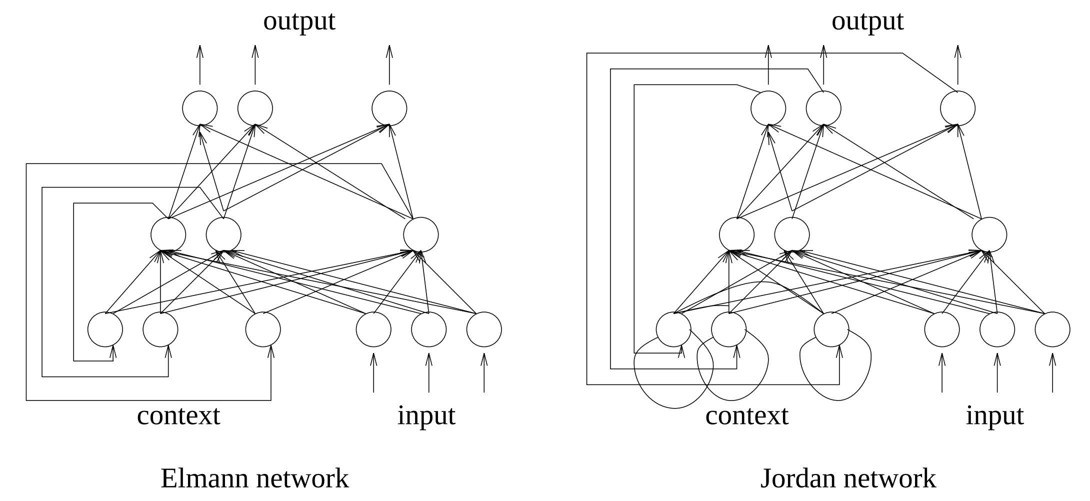


Figure 7.2: Partially recurrent networks.

Lisa Meeden showed how recurrent networks can be used to plan actions of a robot [117]. Stefan Wermter used modules of recurrent networks to implement a natural language parser [220]. Risto Miikkulainen implemented a natural language understanding system [122]. He used modules of feed forward networks and topology preserving maps[4] which formed a recurrent connected module.

7.2.3 Expert networks

An inference network is constructed from the knowledge base of an expert system. The rules of the expert system are directly mapped onto the structure of the neural network [98, 97]. The connection between units directly reflects the rules and the network forms an acyclic directed graph. The computation corresponds to spreading activation from the input units to the output units, but only the input units allow the entry of knowledge. Certainty factors and their algebra are used

[3]A hidden layer is the layer between the input layer and the output layer.

[4]A topology preserving map represents a high-dimensional input data on a low dimensional, usually 2-D topological map [85].

to represent the uncertainty. These certainty values are learned from examples and are represented by the connection values. The units of the neural network do the corresponding calculations. Training consists of a set of input/output values. Wrong CF values converge during training to correct CF values. Here is the previous example, in this instance, however, with certainty factors:

- If flies(x) ∨ feathers(x)) ∧ lays eggs(x) then bird(x) (with CF 1.0)
- If bird(x) ∧ swims(x) then penguin(x) (with CF 0.5)
- If bird(x) ∧ sings(x) then nightingale(x) (with CF 0.7)

is represented as an expert network in fig 7.3.

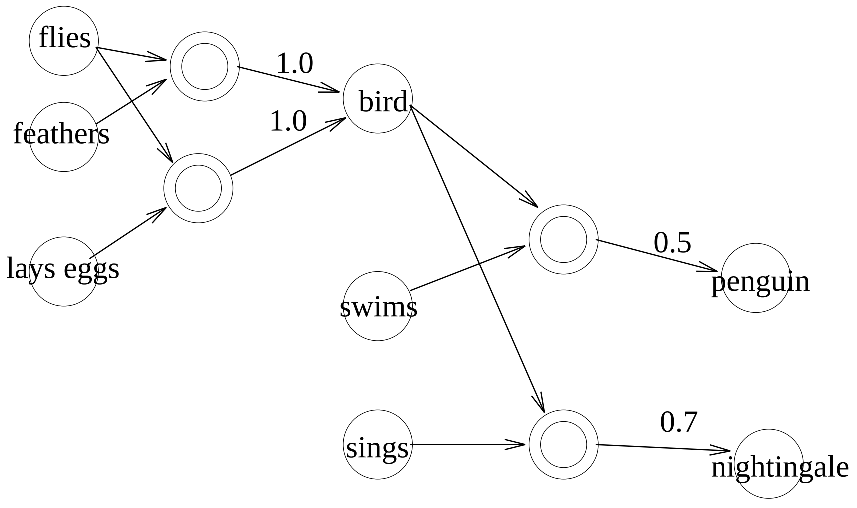


Figure 7.3: Expert network.

Experiments were performed with a wine advisor system which was composed of 44 rules. Expert networks represent a direct map of the symbolic structures onto a neural network structure. The advantages of this approach are the parallel computation and the learning of the CF values from examples.

7.2.4 Neural query-reply system

Hierarchical categorization is related to many neural networks which operate with feature representation. The query-reply system of Anders Holst [73, 72] uses Bayesian neural networks. In this system, only those questions are posed which include the highest information content, as this is the most uncertain feature. A system for categorization of 32 animals was introduced, with each animal defined by approximately 15 features. In our approach this task is realized by a taxonomical arrangement of the knowledge.

7.2.5 First-order predicate calculus neural network

Reflexive reasoning is modeled by a neural network [174, 69, 173]. Reflexive reasoning describes rapid and spontaneous reasoning without conscious effort. The neural network emerges by the mapping of the first-order predicate calculus sentences to a directed acyclic graph. The binding of objects to variables is performed by synchronicity. The nodes represent variables and objects, the connections the conclusions. The reflexive reasoning is modeled by the propagation of the bindings of the objects to the variables through out the network. The system is also able to represent restricted first-order predicate calculus sentences.

7.2.6 Dual representation

A two-level neural system is used [198, 197]. A localistic representation is used in the first level and a distributed in the second. The first level represents rules by an acyclic directed graph. The precondition and the conclusion of the first level correspond to concepts. These concepts are linked to the features of the second level, which are fine grain elements. The features of the second level form rules which are also represented by an acyclic directed graph. The two level architecture allows the representation of uncertainty. During reasoning the rules of the first level are executed first and corresponding distributed representation of the concepts by features are activated. In the next phase, the distributed rules of the second level are executed. The new activated features of the second level activate with a different degree of strength the units of the first level which represent the concepts. These activated concepts represent the result.

In a newer model [199, 200], the second level is replaced by a neural network which performs reinforcement learning. In a task a robot learns to navigate through a minefield. In the first level, rules describe the possible moves. These rules enable a faster reinforcement learning of the second layer, compared to the reinforcement learning without this additional knowledge.

7.2.7 Neural Production systems

DSCP

DSCP is the distributed connectionist production system [208, 209]. A production rule of a DCPS consist of a premise and a conclusion. A premise is described by two triples and these triples are matched against the working memory. A conclusion consists of commands for adding and deleting triples of the working memory. The rules are represented by a subpopulation of units in the rule space. The working memory is also represented by a subpopulation of units in the working memory space. The system operates as a Boltzman machine, finding a rule that matches the state of the working memory by simulated annealing. After the network has settled into a low energy, indicating a consistent rule match, the changes to the working memory are made according to the rule. The execution continues with another simulated annealing process, resulting in another rule application. No backtracking or learning is performed in this model.

Neuro-Soar

Neuro-Soar is an attempt to show that the system SOAR can be implemented by neural networks [30]. The atomic part of the model is the graph representation which is interpreted as a neural network. Objects are represented by graphs, binding by edges of the graph. Productions are also represented by graphs, just like the elements of working memory. A production fires if the graph representing the precondition, and the graph representing the working memory elements match together. However, there is no possibility in this model to represent the problem space or to perform a search.

7.2.8 Neuro solver

Each possible state of a certain domain is represented by a unit [17, 16]. All units are connected together according to the possible state transitions between those states. A breadth-first search in parallel can be performed in this manner. The unit indicating the initial state is activated. The activation spreads out through the network and if the activation meets a unit representing the desired state, a corresponding state sequence is found.

7.2.9 Neural Assembly models

Thomas Wennekers described the simulations of finite automaton by synfire chains which are closely related to the assembly concept [219]. The simulation of finite automaton by neural networks [219] is also referred to in the control module of our associative computer model.

Reasoning with assemblies is described by the "pump of thougths" model. Thought pump is a theoretical assembly model in which thoughts are propagated and changed by different assemblies [23, 25]. Palm substantiated this model [143, 144]. In this model, the associative memory performs the computation of an assembly. Thoughts correspond to situations and the associative memory associates to a situation a list of possible actions which lead to new situations, along with the relative merit of each action. These associations with the evaluation values are then learned by examples. A suggested model could be realized through pattern completion performed by the associative memory. Only an initial situation is given, and the resulting pattern which emerges represents the possible actions with evaluation values [143]. Intelligent input/output coding is crucial for a reasonable performance of this model [143]. Parts of this model were introduced in the section about associative prediction.

Our approach is highly influenced by the assembly theory. It differs mainly in the separation of representation of associations by a permutation associative memory and the evaluation values which result from the pattern or prediction heuristic. The permutation associative memory enables the representation of the resulting associative states. The learning from experience is done separately by the presiction associative memory. Through this separation, the computation does not collapse when the predicted states contain errors.

7.3 Conclusion

It was shown that symbolic reasoning can be performed by neural networks. The dual representation model used two levels of knowledge which allowed the representation of uncertainty. Hidden Markov models pose a very promising and new approach to problem solving. Reinforcement learning and Q-learning demonstrated the abilities of learning from examples, Q-learning primarily in stochastic domains. The neuro solver introduced a model for parallel problem solving if all states are known in advance. Neural assembly models are psychologically and biologically motivated. They describe the learning from experience by assemblies of neurons. The assemblies are simulated by associative memories. The associative computer is a working model of the abstract biological and psychological pump of thougths. It can deal with information that is fuzzy, probabilistic or noisy, it can learn and it is not constrained to only one problem domain.

Chapter 8

Conclusion

Models of neural networks which simulate human problem solving capabilities were examined. Ideas from the artificial intelligence field concerning the treatment of the problem space combined with ideas on neural networks utilizing internal distributed representation led to a fruitful union. A neural model for a deduction system based on the assembly theory was introduced. It was shown that hierarchical categorization can be efficiently performed by neural networks. Based on the previous model, a neural model for a reaction system based on the assembly theory was introduced: the associative computer. This biologically inspired model shades some light on how some problem solving abilities might actually be performed by the human brain. It also highlights the benefits of distributed representation. These benefits include the ability to learn from experience, heuristics resulting from picture representation, the ability to deal with noisy information, and the ability to do some work in parallel. The description of the reasoning process by formation of associations represents a direction in which different fields of study meet together.

8.1 Summary of central principles and contributions

Associative memory is, despite its simplicity, an efficient and biologically plausible model which captures the basic behavior principles of the cell assembly theory. It is the basic building block of the succeeding models. An intital model composed of associative memories which performs associative categorization was introduced. The reliability of the answer was indicated by the defined quality criterion.

Categorial representation Verbal categorization was introduced. Uncertainty is identified with noise resulting from fault or fragmentary pattern representation. The quality criterion representing the rating in the presence of a verbal category is based on counting which is performed by units of the associative memory. Consequently, there is no division between knowledge processing and an uncertainty algebra. The table of belief values enables the separation of the uncertainty of the expert who defines the knowledge base from the observer who uses the system.

Visual representation by cognitive entities is motivated by the "what" and "where" system of the brain. Picture representation by the cognitive entity enables a solution of the binding problem. Associations can be defined which can handle the uncertainty by the similarity principle. The cognitive entity includes the holistic representation by patterns which neglects the modular structure. Pattern representation enables the definition of the quality criterion that rates the value of different states according to their similarity to a visual category.

Hierachical categorization A neural model for a deduction system based on the assembly theory was introduced. The system consists of several neural associative memories which are organized in a hierarchical way. The problem space is represented by static connections between them. The knowledge is represented

Figure 8.1: Neural deductions system.

by a taxonomic arrangement of verbal categories. An availabity heuristic was defined which models the effects of priming emphasis and forgetting. An availability heuristic offers a combination of the frequency with the actual likelihood of the presence of a verbal category. The model learns to favor those categories which often lead to a successful goal and this may help to speed up the search. Three applications were presented. The representation and access of large knowledge bases was demonstrated by the system Jurassic which helps the paleontologist to determine creatures from uncertain knowledge. Experiments with the availability heuristic in which parts of the knowledge base were primed or forgotten were performed.

Associative Computer A neural model for a reaction system based on the assembly theory was introduced. The system uses picture representation to perform planning instead of symbolic representation. Picture representation allows the presence of noise and also enables learning from examples. Cognitive entity representation of pictures allows the recognition and execution of associations with the permutation associative memory. The permutation associative memory solves two important problems which arise in the domain of neural networks. First, the traditional associative memory can not learn several possible associations which arise from a single input. The second problem is the binding problem which determines how to connect all separated fragments of a complex object.

- The retrieval phase of the permutation associative memory corresponds to the determination of all possible instantiations of rules in a symbolic production system.
- The retrieval phase is subdivided into two stages: the attention stage [5, 155] and the binding stage. Only when the attention is focused on certain parts are they bound to a whole object [210, 34, 155] representing a question.
- The binding problem is solved after the attention stage by constraining the formation of permutations of parts of an associative memory.

The permutation associative memory, the search chain and the controller form the associative computer which models biologically plausible computation in accordance with the pump of thoghts idea (see fig. 8.2).

Picture representation leads to the pattern heuristic which speeds up the search. The solved problems were used to speed up the search of related problems significantly. Due to picture representation learning from examples by an associative memory could be performed. After learning the associative memory guided the search. Different learning strategies were examined.

Experiments with the associative computer It was shown that the used representation is not constrained to a special domain through examples of three different domains. It was demonstrated that the pattern heuristic and the prediction heuristic sped up the search in problem solving significantly in different domains. Because of these empirical results, two hypotheses were claimed. The

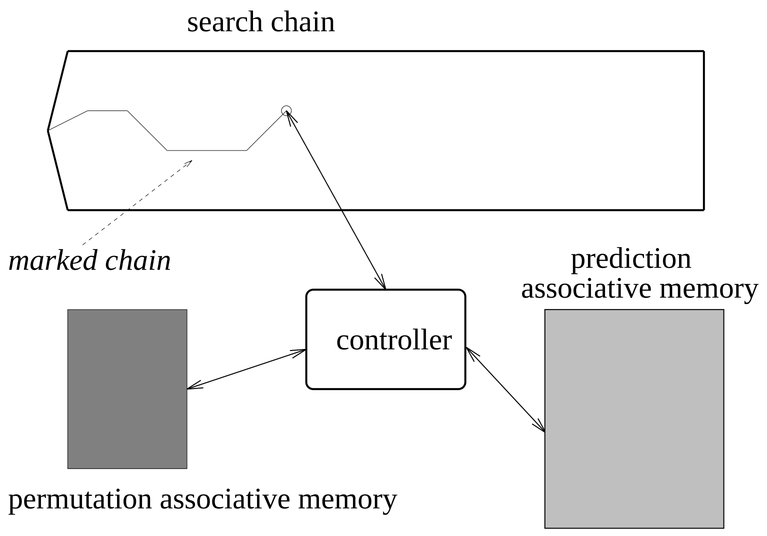


Figure 8.2: The permutation associative memory, the search chain and the controller form the associative computer. The controller links the permutation associative memory, the prediction associative memory and the search chain. It controls the computation.

best results were obtained with supervised learning with an observer who chose relevant examples which described the problems as well as possible.

8.2 Implications

The associative computer uses picture representation. The similarity between pictures can be computed. Because of this property, a far reaching interpretation of the idea symbol is needed in the physical symbol system hypothesis by Newell and Simon (1976) [135]. It was shown that picture representation leads to a simplification in the simulation of some human intelligence behavior in machines.

It is often helpful to use picture representation which orients itself on the real world when human intelligence behavior is simulated. The elimination of this additional information is not helpful.

This predication is motivated by interpretation of uncertainty with noise in the pattern representation and by the three theses which sustain themselves by the experiments presented in this work.

Thesis 1 *Shape similarity between representation of states through visual categories is significantly similar to the distance in the problem space.*

Thesis 2 *Representation of states through visual categories enables the access of knowledge which was formed by learned experience during problem solving.*

Thesis 3 *An adequate model of human thought uses the flexibility of similarity-based inference and the compositionality and certainty of rule-based inference.*

The third these, motivated by the associative computer architecture, gives an answer to the ongoing rule- versus similarity-based processing debate in cognitive science [61, 188].

8.3 Looking ahead

The permutation associative memory could also be used in other domains. Certain structures in sequences could be recognized, for example, structures in DNA sequences.

Additional experiments with the associative computer could be performed:

- Asynchronous unsupervised learning could lead to improvement in the results. Examples which are learned are chosen randomly and not in a fixed order.
- The performance of the associative computer with objects of different size could be examined. Objects of not-learned size would by treated as "noisy" patterns by the permutation associative memory. This is because the size is indicated by two bars.
- Representation of three dimensional visual states. (Higher dimensoins are also possible.)
- Different similarity measures could be examined.
- Improved bi-directional retrieval of the associative memory could be used [190].
- Mathematical analysis of the behavior of the prediction associative memory during learning (difference equations).

8.3.1 Search

Two modifications corresponding to the representation of the problem space could be performed.

One modification is the principle of ignoring the local spreading activation values which are propagated by links in the search chain, and instead implementing a best-first search. A best-first [229, 166, 111] search does not get stuck in local maxima. By a suitable definition of the evaluation function $f(n) = g(n) + h(n)$ algorithm A is obtained (n is any state encountered in the search, $g(n)$ the cost of n from the initial state and $h(n)$ a heuristic estimate). If $h(n) \leq h^*(n)$ where $h^*(n)$ is the minimal cost to the desired state, A^* algorithm is obtained. A^* algorithm always finds the best solution (shortest plan length) [229, 166, 111].

The other possibility is the reduction of the memory requirements at the price of impairing the search. The sequence of visited states is stored in an associative memory in the same way that the sequence describing a plan is stored in the prediction associative memory of the associative computer model (see fig. 5.21). By comparison with these states, loops can be prevented. However, because of the possibility of errors in reconstructing previous states, loops can emerge and backtracking is not always possible.

8.3.2 Speculations

The examples of introduced problems indicate that the associative computer is qualified for usage in an autonomous robot [10].

It would be useful, however, to have a preprocessing phase. Pictures of a camera could be preprocessed into (visual-categorical) representation by the cognitive entities. This could be done by neural networks such as the Neocognitrom [49] or map transformation cascades [222].

Learning of associations could then be performed. From corresponding changes in the succeeding pictures supplied by a camera, associations could be extracted. Before the learning, a test could be performed to verify if those associations were not already learned before (see paragraph about knowledge revision in chapter 2).

The generalization of the learned associations could be performed by a direct manipulation on the weight matrix of the permutation associative memory. Pattern representation of pictures would lead to ordered weight matrixes. The order would enable a direct manipulation of particular zones of the weight matrix. This could be done by a meta-neural network which examines and changes the weight matrix of the permutation associative memory.

Finally, the associative computer could be integrated in a hierarchical planner with the ability of chunking.

8.4 Associative Computation

In this work we examined how human problem solving can be modeled by neural networks. We tried to ask the central question of whether or not neural systems increase our understanding of cognition.

Problem solving is an essential domain for understanding intelligence. The Artificial Intelligence field has tried to model human intelligence behavior and the production system model is an important model within the field which describes problem solving. It is also utilized by some psychologists as a model of human cognition.

Another idea is to describe the human reasoning process as the formation of associations. There are many ideas as how to do it, but there is no sufficiently detailed explanation of how those ideas might work. No complete model exist which could be examined by computer simulations. Anderson writes in addition [7], page 581:

"Associations can give flexibility and power in operation; however, the proper way to use it, the proper data representations, and the proper network architecture are still not known. We think that this area is one of the frontiers of the cognitive science."

As recently as in August 1998 Dave Touretzky asked on the connectionistic e-mailing list: "Is connectionist symbol processing dead?"

From ml-connectionists-request@mlist-1.sp.cs.cmu.edu Tue Aug 11 17:35:10 1998
From: Dave_Touretzky@cs.cmu.edu
To: connectionists@cs.cmu.edu
Subject: Connectionist symbol processing: any progress?
Date: Tue, 11 Aug 1998 03 : 34 : 27 *-0400*

"I'd like to start a debate on the current state of connectionist symbol processing? Is it dead? Or does progress continue? ... People had gotten some interesting effects with localist networks, by doing spreading activation and a simple form of constraint satisfaction.... This approach does not create new structure on the fly, or deal with structured representations or variable binding. Those localist networks that did attempt to implement variable binding did so in a discrete, symbolic way that did not advance the parallel constraint satisfaction/heuristic reasoning agenda of earlier spreading activation research. ... So I concluded that connectionist symbol processing had reached a plateau, and further progress would have to await some revolutionary new insight about representations. ... The problems of structured representations and variable binding have remained unsolved. No one is trying to build distributed connectionist reasoning systems any more, like the connectionist production system I built with Geoff Hinton..." (see chapter 7, DSCP)

After an exchange of ideas it was clear that no revolutionary new insight about representations was presented and that no one participating in the discussion is currently trying to build distributed connectionist reasoning systems [77]. The present work tries to fill the explicatory gap. This is done by an evolutionary approach in which ideas from Artificial Intelligence and the ideas from neurocomputing lead to a new model, the associative computer.

8.4.1 Summary

Associative categorization which relied on the associative memories was introduced. Already even in this uncomplicated application, the main advantages of associative computation principles were demonstrated, namely fast computation linked with toleration of errors.

Categories were shown to be the basic building blocks of knowledge representation when human problem solving is simulated by neural networks. Distributed representation and similarity are the natural properties of categorical representation and it is these properties which distinguish categorial representation from symbolical representation.

Similarity is used when uncertain knowledge is represented, without the need of an additional calculus. In addition belief tables allow the detachment of the uncertainty of the coded knowledge and the actually present knowledge.

Priming during hierarchical categorization eases the formation of hypothesis, as more exact, possible hypothesis are formed. The human behavior of emphasizing knowledge areas which are often used and simultaneously disregarding of knowledge areas which are seldom used was modeled. This policy is useful, if one has to balance between completeness and speediness of retrieved knowledge.

The associative computer has the same behavior as a pure symbolical reaction system (production system). It was shown how a mapping of a symbolical reaction system to a neural network can be performed. For a cognitive-science scientist it is important to understand how such computations can be performed by the human brain.

An important research field in Artificial Intelligence is the usage of heuristics to speed up the search. It is not easy to define heuristic functions, as there is no rule which says how to do this. A heuristic function results automaticaly in the associative computer from the representation of knowledge by pictures. Another approach to the view of heuristic functions is shown: heuristic functions which result from the manner of description of the knowledge. The first intuitive idea, that a state represented as a picture is nearer to a desired state represented as a picture, the more similar those pictures are, was confirmed statistically. In fact a pattern heuristic speeds up the search. The real world give us insights on how to solve the problem. Such an insight is used by the pictorial representation and the resulting similarity criterion. This information is not always correct, but it is generally better to obey it than to ignore it.

Another property of the associative computer is the possibility of learning with a teacher. A teacher who chooses pertinent examples of a domain and another teacher who demonstrates how to solve those problems can help the associative computer in learning, and in efficiently solving problems in this domain. Faulty information which is sometimes indicated by the pattern heuristic can be in this way prohibited. An important condition is the appropriate choice of material. Wrong material, too much, or too little material can worsen the problem solving behavior. A good education about the problem space is essential for fast and effective problem solving. In our example, the pattern heuristic acted as the second teacher.

The associative computer learns from experience if no additional information from a teacher is available. It learns to prohibit faults. This kind of learning is tedious, and different problems indicate contradictory recommendations. Despite the low load of the prediction associative weight matrix, an overlearning occurs. This is at first glance a contradiction, but by closer examination one recognizes that important information is concentrated in a small area of the prediction associative memory. This is because during problem solving, the world changes only minimally. Those changes are represented in the prediction associative memory and are concentrated in a small area of the weight matrix. Many more patterns are stored than during the supervised learning. Some learning sequences are needed until the stabilization of the behavior of the associative computer. After the first learning session, an impairment is present while an improvement is present in the succeeding steps. The improvement comes along with the gain followed by the reduction of the weights of the prediction associative memory. The behavior of the associative computer after the learning is comparable to the pattern heuristic.

The information supplied by sensors during problem solving can often be "noisy" or incomplete. Noise can also result from faulty hardware. For example, the cameras of a robot can supply noisy pictures, such as when the sensors of a space probe have a malfunction. In this case, it is important that the computing system does not collapse. It was shown that the associative computer is a robust model and can tolerate noise to some extent. It is an ideal model for performing problem solving computation in a robot. On the other hand every biologically plausible model which simulates human problem solving behavior performed by the brain should tolerate noise to some extent, as noise is present in the real human brain.

A model which attempts to explain the process of human problem solving must not be constrained to only one area. It should have the same behavior in different domains. Exemplary domains which are well known and extensively studied in Artificial Intelligence were examined. It was shown how those domains can be represented with the aid of cognitive entities. It was also shown that the associative computer has the same behavior in those domains. It tolerates noise, learns, and uses heuristic information resulting from representation. The choice of the appropriate example as teaching material is very important. The prediction associative memory which was formed by learning the Sussman anomaly describes the problem of building the CBA tower statistically better than the pattern heuristic. One well-chosen example catches all the essential information in about the nature building the CBA tower. Unsupervised learning of many problems is difficult. This is because different states describing problems differ only slightly. Despite this fact, the prediction associative memory extracts the relevant knowledge which leads to a significant improvement in the problem solving behavior. The examined domains of problems indicate that the associative computer is qualified for usage in a robot.

8.4.2 Epilogue

The human ability to process images and understand what they mean in order to solve a problem holds an important clue to how the human thought process works. This clue was examined by empirical experiments with the associative computer. One general conclusion from the experiments is the claim that it is possible to use systematically associative structures to perform reasoning by forming chains of associations. In addition, beside symbolical problem solving, pictorial problem solving is possible.

Appendix

Appendix A

A.1 Statistical tools

A.1.1 Hypothesis testing

It is assumed that the modified model has the same performance as the unmodified one. This is called the *null hypothesis*. If this is not true then the *null hypothesis* is wrong. Statistical hypothesis testing bounds the probability of incorrectly asserting that the *null hypothesis* is false [33].

A.1.2 The t test

The sampling distribution of the mean approaches the normal distribution as the sample size approaches infinity. If the sample size n is small, different distributions result depending on the sample size n. The t test depends on the assumption that the distribution from which the sample was drawn is normal. However, unless n is very small and the distribution is skewed and the derived score is marginal, it can be trusted [33]. t is a family of distributions, one for each n

$$t = \frac{\overline{x} - \mu}{s/\sqrt{n}}$$

with:

- $\overline{x}$ mean of this sample.
- μ mean value.
- s standard deviation of this sample.
- n sample size.
- $n - 1$ degrees of freedom.

The *null hypothesis* assumes that the mean value is the same as the mean value of the sample. It is not the same if it is significantly different.

A.1.3 p Value

The probability of obtaining a particular sample result given the null hypothesis is called a p value. By convention the hypothesis is rejected if $p < 0.05$. This is the bound on p which is considered statistically significant. p value can be determined by given degrees of freedom and the t values from a table of t distributions or by a statistic computer program.

A.1.4 t statistic for two means

Given two random samples $x_1, x_2, \cdots, x_{n1}$ and $y_1, y_2, \cdots, y_{n2}$ from normal populations with unknown means μ_1, μ_2 and the unknown variance σ^2. The *null hypothesis* is:

$$H_0 : \mu_1 = \mu_2.$$

It can be tested with the t statistic

$$t = \frac{\overline{x} - \overline{y}}{\sqrt{\frac{1}{n_1} + \frac{1}{n_2}} \cdot \sqrt{\frac{\sum x_i^2 - n_1\overline{x}^2 + \sum y_i^2 - n_2\overline{y}^2}{n_1+n_2-2}}}$$

with the t distribution with $n_1 + n_2 + 2$ degrees of freedom.

A.1.5 Paired t statistic

A paired sample t test is used when two strategies are used on the same set of problems. For example, a progam is run with and without a certain strategy on the same data and the difference in performance is recorded. Given a set of paired observations $x_1, x_2, \cdots, x_n$ and $y_1, y_2, \cdots, y_n$ from normal populations with unknown means μ_1, μ_2, the *null hypothesis* is:

$$H_0 : \mu_1 = \mu_2$$

It can be tested with the t statistic

$$t = \frac{\overline{D}}{s_D} \cdot \sqrt{n}$$

with the t distribution with $n - 1$ degrees of freedom. The variable definitions are:

$$D_i = x_i - y_i$$

$$\overline{D} = \frac{1}{n} \sum_{i=1}^{n} D_i$$

$$s_D = \sqrt{\frac{\sum D_i^2 - \frac{1}{n}(\sum D_i)^2}{n - 1}}.$$

A.1.6 Error Bars

Scale is arbitrary in plots of mean values over time. To suppress the scale effects in the plot, error bars are added to the plot of means [33]. Each bar represents an upper and lower bound of a 95 percent confidence. The 95 percent confidence interval corresponding to the error bar is for μ with t_p, $p = 0.025$ and $n - 1$:

$$\overline{x} \pm t_{0.025} \cdot s/\sqrt{n}.$$

A.2 Object oriented laboratory

The hierarchy of the complexity of the models is reflected by the classes they use (see fig. A.1) [119, 22].

1. The basic classes which form the atomic part is the class which represents the *binary vector* in the pointer format. The second atomic class implements the *associative memory* [221].

2. The *word recognition* class is composed from both atomic classes. It enables the implementation of associative categorization.

3. The *word recognition* class and the *associative memory* class allows the definition of the *module* class. From these classes the hierarchical categorization architecture is composed. The problem space is implemented by connections between the modules.

4. The associative computer model uses dynamical representation of the problem space. It uses one class *permutation associative memory* which is made of the *associative memories*. The *cognitive entity* representation class uses the *binary vector* representation. The techniques of the problem space orient themselves on the system for hierarchical categorization.

This section describes where central information concerning the implementation of the associative computer is located. The associative computer is composed of the permutation associative memory, the search chain, the prediction associative memory and the controller.

- First representation of states and associations should be implemented. In section 3.4. representation of states and associations is described.

- After that associative memory should be implemented. In chapter 2 the principles of associative memory are described.

- The principles of the permutation associative memory are described in section 5.3.1.

- The Associative computer is composed of the search chain and the controller which links the permutation associative memory with the search chain. The search chain is described in section 5.3.2. The controller is described by a deterministic automaton in section 5.3.3.

- The additional properties of the associative computer result from the pattern heuristic described in section 5.3.5 and from the prediction heuristic. The prediction heuristic is learned by experience and is described in the section 5.3.6.

Algorithm for the retrieval phase of the permutation associative memory:

```
DO
$jota = 0; Nr = 0$
**recursion:**
ita(jota)=0
**forloop:**
$if((qc_{part} = part(jota)_{ita(jota)}) > threshold_{part})$
   DO
   $if(jota \neq (delta - 1))$
     DO
     jota=jota+1; goto **recursion**
     OD
   $else$
     DO
     flag=0;
     $for(i = 0; \neq \delta - 1; i++)$
         DO
         $for(j = i + 1; \neq \delta; j++)$
             DO
             $if(ita[i] == ita[j])$   flag=1; break
             OD
         $if(flag == 1)$ break
         OD
     $if(flag == 0)$
         DO
         if($(qc_{whole} = \sum_{\iota=0}^{\delta-1} part(\iota)_{ita(\iota)}) > threshold_{whole}$)
             DO
             $state(Nr) = \sum_{\iota=0}^{\delta-1} part(\iota)_{ita(\iota)}$
             Nr=Nr+1
             OD
         OD
     OD
   OD
   **ret:**
   $if(ita[jota] \neq (\Delta - 1))$   ita[jota]=ita[jota]+1; goto **forloop**
   $if(jota \neq 0)$   jota=jota-1; goto **ret**
OD
```

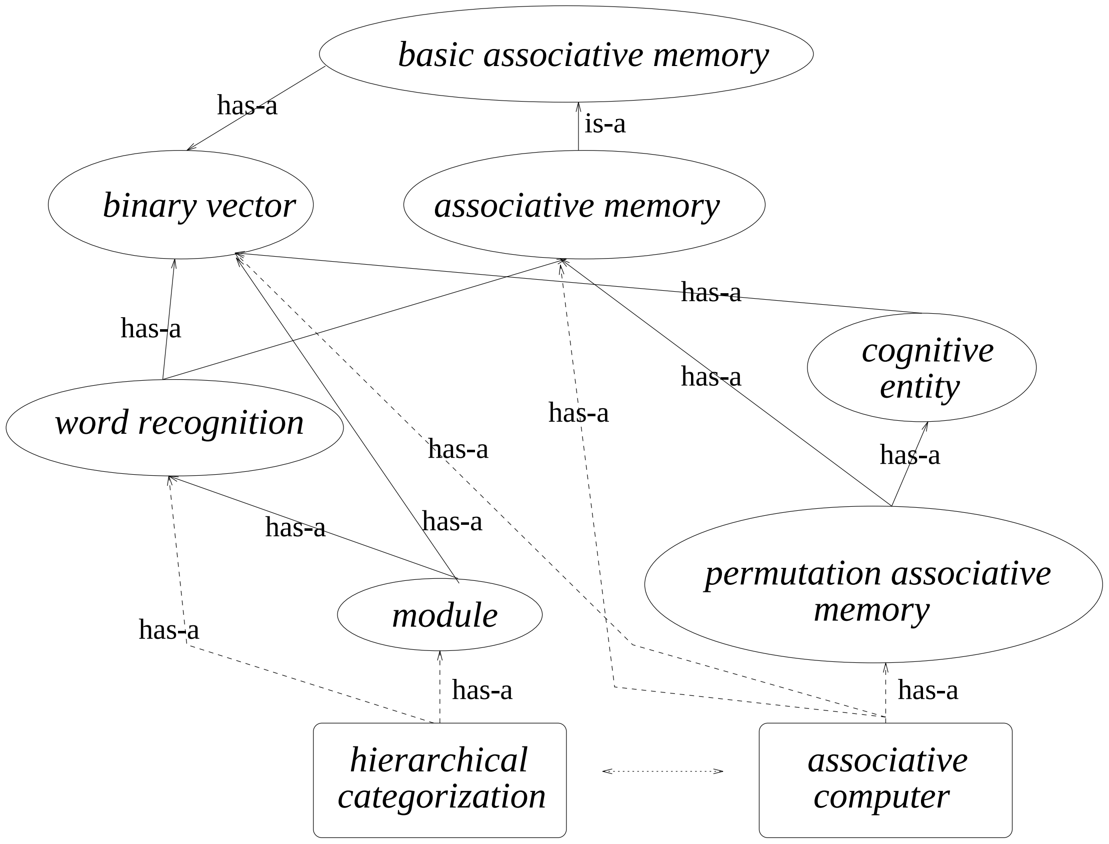


Figure A.1: Class hierarchy in the object oriented laboratory.

A.3 Hierarchical Categorization in System Jurassic

A.3.1 Maiasaura

The species of fig. 4.13 is described by six features using the belief table.

1. probably **bird hipped**
2. certainly **two legged**
3. probably **long arms**
4. certainly **long stiffly held tail**
5. probably **solid bony humps or crests**
6. very probably **length nine m**

```
*
1) ! MODULE ORNITHISCHIAN with qc=0.33, <1> !*
2) ! MODULE ORNITHOPODS with qc=0.67, <2> !**
3) ! MODULE PACHYCEPHALOSAURIDS with qc=0.51, <3> !
4) ! MODULE THESCELOSAURIDS with qc=0.49, <3> !
5) ! MODULE HADROSAURIDS with qc=0.48, <3> !*
6) ! MODULE HADROSAURINE_DUCKBILLS with qc=0.36, <4> !*
---------------------------------------------
R E S U L T :
```

MAIASAURA with qc=0.38
SAUROLOPHUS with qc=0.33

The first determined categoriy represents *Maiasaura* (see fig. 4.13). *Maiasaura* and *Saurolophus* belong to the group *Hadrosaurine duckbills*. The search is continued according to the barrier [1].

```
7) ! MODULE LAMBEOSAURINE_DUCKBILLS with qc=0.3, <4> !*
---------------------------------------------
R E S U L T :
```

JAXARTOSAURUS with qc=0.33
HYPACROSAURUS with qc=0.28

[1] $\omega = -0.5, \varpi = 0.2$.

```
8) ! MODULE HYPSILOPHODONTIDS with qc=0.46, <3> !
9) ! MODULE FABROSAURIDS with qc=0.43, <3> !
10) ! MODULE HETERODONTOSAURIDS with qc=0.42, <3> !
11) ! MODULE IGUANODONTIDS with qc=0.41, <3> !*
---------------------------------------------
R E S U L T :
```

IGUANODON with qc=0.41

The category which corresponds to *Iguanodon* has the highest *qc* value. It is, however, determined later because the presence of the group *Iguanodontits* is less possible then the group *Hadrosaurids*. *Maiasaura* represents the *qc* local maximum, *Iguanodon* the global maximum. Local maximum can emerge during hill climbing search strategy.

```
12) ! MODULE CAMPTOSAURIDS with qc=0.4, <3> !
13) ! MODULE TROOEDONTIDS with qc=0.39, <3> !

14) ! MODULE THEROPODS with qc=0.42, <2> !*
15) ! MODULE COELUROSAURS with qc=0.28, <3> !
16) ! MODULE CAENAGNATHIDS_ELMISAURIDS with qc=0.21, <4> !
17) ! MODULE OVIRAPTORIDS with qc=0.21, <4> !
18) ! MODULE SEGISAURIDS with qc=0.21, <4> !
19) ! MODULE COMPSOGNATHIDS with qc=0.21, <4> !
20) ! MODULE DROMAEOSAURIDS with qc=0.19, <4> !
21) ! MODULE GARUDIMIMIDS with qc=0.19, <4> !
22) ! MODULE ORNITHOMIMIDS with qc=0.19, <4> !
23) ! MODULE SAURORNITHOIDIDS with qc=0.19, <4> !
24) ! MODULE ARCHAEOPTERYGIDS with qc=0.18, <4> !
25) ! MODULE COELOPHYSIDS with qc=0.18, <4> !
26) ! MODULE COELURIDS with qc=0.17, <4> !
27) ! MODULE NOASAURIDS_SHANSHANOSAURIDS with qc=0.16, <4> !
28) ! MODULE DEINOCHEIRIDS with qc=0.16, <4> !
29) ! MODULE AVIMIMIDS with qc=0.13, <4> !

30) ! MODULE CARNOSAURS with qc=0.24, <3> !*
31) ! MODULE ALLOSAURIDS with qc=0.17, <4> !
32) ! MODULE TERATOSAURIDS with qc=0.17, <4> !
33) ! MODULE SEGNOSAURIDS with qc=0.16, <4> !*
```

```
----------------------------------------------
R E S U L T :
```

SEGNOSAURUS with qc=0.22

```
34) ! MODULE SPINOSAURIDS with qc=0.16, <4> !
35) ! MODULE CERATOSAURIDS with qc=0.15, <4> !
36) ! MODULE DRYPTOSAURIDS with qc=0.15, <4> !
37) ! MODULE TYRANNOSAURIDS with qc=0.15, <4> !
38) ! MODULE MEGALOSAURIDS with qc=0.15, <4> !*
----------------------------------------------
R E S U L T :
```

POEKILOPLEURON with qc=0.2
MEGALOSAURUS with qc=0.2

```
39) ! MODULE OTHER with qc=0.1, <4> !
40) ! MODULE THERIZINOSAURIDS with qc=0.1, <4> !

41) ! MODULE STAURIKOSAURIDS with qc=0.11, <3> !

42) ! MODULE FOUR_LEGGED_ORNITHISCHIANS with qc=0, <2> !
43) ! MODULE CERATOPSIANS with qc=-0.02, <3> !
44) ! MODULE PSITTACOSAURIDS with qc=-0.05, <4> !
45) ! MODULE PROTOCERATOPSIDS with qc=-0.05, <4> !
46) ! MODULE SHORT_FRILLED_CERATOPSIDS with qc=-0.06, <4> !*
47) ! MODULE LONG_FRILLED_CERATOPSIDS with qc=-0.07, <4> !

48) ! MODULE STEGOSAURIDS with qc=-0.05, <3> !*
49) ! MODULE ANKYLOSAURS with qc=-0.06, <3> !
50) ! MODULE NODOSAURID_ANKYLOSAURS with qc=-0.09, <4> !
51) ! MODULE ANKYLOSAURID_ANKYLOSAURS with qc=-0.09, <4> !

52) ! MODULE SCELIDOSAURIDS with qc=-0.08, <3> !

53) ! MODULE SAUROPODOMORPHS with qc=-0.05, <2> !
54) ! MODULE PROSAUROPODS with qc=-0.07, <3> !
55) ! MODULE PROSAUROPOD_ODDITIES with qc=-0.05, <4> !
56) ! MODULE HERRERASAURIDS with qc=-0.07, <4> !
57) ! MODULE ANCHISAURIDS with qc=-0.11, <4> !
```

```
58) ! MODULE PLATEOSAURIDS with qc=-0.12, <4> !
59) ! MODULE ROCCOSAURIDS with qc=-0.14, <4> !

60) ! MODULE SAUROPODS with qc=-0.07, <3> !
61) ! MODULE DIPLODOCIDS with qc=-0.07, <4> !
62) ! MODULE CAMARASAURIDS with qc=-0.08, <4> !
63) ! MODULE CETIOSAURIDS with qc=-0.08, <4> !
64) ! MODULE TITANOSAURIDS with qc=-0.1, <4> !*
---------------------------------------------
R E S U L T :
```

ALGOASAURUS with qc=0.01

```
65) ! MODULE BRACHIOSAURIDS with qc=-0.11, <4> !

66) ! MODULE ASSORTED_SAUROPODS with qc=-0.19, <3> !

67) ! MODULE SAURISCHIAN with qc=-0.33, <1> !*
68) ! MODULE STRANGE_KILLERS with qc=-0.17, <2> !

sorry, I have no more acceptable answer for You.....
```

A.3.2 Stenonychosaurus

Suppose we pose a question: Which dinosaur species is most similar to a human? We describe a human being by seven features. The hierarchical categorization is performed in which the most similar stored category is determined.

1. certainly **two legged**
2. certainly not **four legged**
3. probably **thin walled fragile bones**
4. very probably not **big**
5. very probably **big eyes**
6. certainly **big brain**
7. certainly **capsule in the skull**
8. very probably **length two m**

```
1) ! MODULE SAURISCHIAN with qc=-0.33, <1> !**
2) ! MODULE THEROPODS with qc=0.05, <2> !*
3) ! MODULE COELUROSAURS with qc=0.12, <3> !***
4) ! MODULE SAURORNITHOIDIDS with qc=0.2, <4> !*
----------------------------------------------
R E S U L T :
```

SAURORNITHOIDES with qc=0.25
STENONYCHOSAURUS with qc=0.25

This answer is the same as the suggestion of Dale Russell [165] in the early 1980s that the Stenonychosaurus, see fig 4.14 (also now known as Troodon), could have given rise to a brainy descendant, had dinosaurs survived instead of dying out [101, 102]. The search is continued without any local maximal.

```
5) ! MODULE DROMAEOSAURIDS with qc=0.14, <4> !*
----------------------------------------------
R E S U L T :
```

ADASAURUS with qc=0.2

```
6) ! MODULE OVIRAPTORIDS with qc=0.12, <4> !
7) ! MODULE GARUDIMIMIDS with qc=0.1, <4> !
8) ! MODULE CAENAGNATHIDS_ELMISAURIDS with qc=0.09, <4> !*
----------------------------------------------
R E S U L T :
```

CAENAGNATHUS with qc=0.16
CHIROSTENOTES with qc=0.16
ELMISAURUS with qc=0.16
MACROPHALANGIA with qc=0.16

```
9) ! MODULE SEGISAURIDS with qc=0.09, <4> !
10) ! MODULE COMPSOGNATHIDS with qc=0.09, <4> !
11) ! MODULE ORNITHOMIMIDS with qc=0.07, <4> !
12) ! MODULE ARCHAEOPTERYGIDS with qc=0.06, <4> !
13) ! MODULE COELOPHYSIDS with qc=0.06, <4> !*
```

```
---------------------------------------------
R E S U L T :
```

LUKOUSAURUS with qc=0.1

```
14) ! MODULE COELURIDS with qc=0.05, <4> !*
---------------------------------------------
R E S U L T :
```

ARISTOSUCHUS with qc=0.13
COELURUS with qc=0.13
COMPSOSUCHUS with qc=0.13
ORNITHOLESTES with qc=0.13

```
15) ! MODULE NOASAURIDS_SHANSHANOSAURIDS with qc=0.04, <4> !
16) ! MODULE DEINOCHEIRIDS with qc=0.04, <4> !
17) ! MODULE AVIMIMIDS with qc=0, <4> !

18) ! MODULE ORNITHOPODS with qc=0.03, <3> !
19) ! MODULE THESCELOSAURIDS with qc=0.01, <4> !
20) ! MODULE PACHYCEPHALOSAURIDS with qc=0.01, <4> !
21) ! MODULE HYPSILOPHODONTIDS with qc=0, <4> !*
---------------------------------------------
R E S U L T :
```

FULGUROTHERIUM with qc=0.08
LONCOSAURUS with qc=0.08

```
22) ! MODULE FABROSAURIDS with qc=0, <4> !
23) ! MODULE HADROSAURIDS with qc=0, <4> !
24) ! MODULE HADROSAURINE_DUCKBILLS with qc=-0.04, <5> !
25) ! MODULE LAMBEOSAURINE_DUCKBILLS with qc=-0.05, <5> !
```

```
26) ! MODULE CAMPTOSAURIDS with qc=-0.01, <4> !
27) ! MODULE HETERODONTOSAURIDS with qc=-0.01, <4> !
28) ! MODULE TROOEDONTIDS with qc=-0.02, <4> !
29) ! MODULE IGUANODONTIDS with qc=-0.02, <4> !

30) ! MODULE CARNOSAURS with qc=-0.01, <3> !
31) ! MODULE SEGNOSAURIDS with qc=-0.02, <4> !
32) ! MODULE SPINOSAURIDS with qc=-0.03, <4> !
33) ! MODULE CERATOSAURIDS with qc=-0.03, <4> !
34) ! MODULE DRYPTOSAURIDS with qc=-0.04, <4> !
35) ! MODULE TYRANNOSAURIDS with qc=-0.04, <4> !
36) ! MODULE MEGALOSAURIDS with qc=-0.04, <4> !
37) ! MODULE ALLOSAURIDS with qc=-0.04, <4> !
38) ! MODULE TERATOSAURIDS with qc=-0.05, <4> !
39) ! MODULE OTHER with qc=-0.09, <4> !
40) ! MODULE THERIZINOSAURIDS with qc=-0.09, <4> !

41) ! MODULE STAURIKOSAURIDS with qc=-0.08, <3> !*
----------------------------------------------
R E S U L T :
```

STAURIKOSAURIDS with qc=0.05

```
42) ! MODULE STRANGE_KILLERS with qc=-0.17, <2> !

43) ! MODULE ORNITHISCHIAN with qc=-0.33, <1> !**

sorry, I have no more acceptable answer for You.....
```

A.3.3 Parasaurolophus

The links between the categories are not initialized and a species from another group is searched. It is *Parasaurolophus* of the "Lambe's lizards" group (see fig 4.18). The tube inside a *Parasaurolophus*'s *crest acted as a sound box, amplifying the voice and producing low, resonant cries* [102], page 150.

1. certainly **short muzzle and curved hollow horn jutting back from the head**

2. very probably **late cretaceous**

```
1) ! MODULE ORNITHISCHIAN with qc=-0.33, <1> !
2) ! MODULE FOUR_LEGGED_ORNITHISCHIANS with qc=-0.33, <2> !
3) ! MODULE STEGOSAURIDS with qc=-0.27, <3> !*
```

This was the learned path to another group. After backtracking the search is continued:

```
4) ! MODULE ANKYLOSAURS with qc=-0.28, <3> !
5) ! MODULE NODOSAURID_ANKYLOSAURS with qc=-0.26, <4> !*
6) ! MODULE ANKYLOSAURID_ANKYLOSAURS with qc=-0.26, <4> !*

7) ! MODULE SCELIDOSAURIDS with qc=-0.3, <3> !
8) ! MODULE CERATOPSIANS with qc=-0.24, <3> !
9) ! MODULE PSITTACOSAURIDS with qc=-0.21, <4> !
10) ! MODULE PROTOCERATOPSIDS with qc=-0.22, <4> !*
11) ! MODULE SHORT_FRILLED_CERATOPSIDS with qc=-0.23, <4> !*
12) ! MODULE LONG_FRILLED_CERATOPSIDS with qc=-0.24, <4> !*

13) ! MODULE SAUROPODOMORPHS with qc=-0.38, <2> !
14) ! MODULE PROSAUROPODS with qc=-0.29, <3> !
15) ! MODULE PROSAUROPOD_ODDITIES with qc=-0.22, <4> !
16) ! MODULE HERRERASAURIDS with qc=-0.24, <4> !
17) ! MODULE STAURIKOSAURIDS with qc=-0.27, <4> !
18) ! MODULE ANCHISAURIDS with qc=-0.28, <4> !
19) ! MODULE PLATEOSAURIDS with qc=-0.29, <4> !
20) ! MODULE ROCCOSAURIDS with qc=-0.3, <4> !

21) ! MODULE SAUROPODS with qc=-0.29, <3> !
22) ! MODULE DIPLODOCIDS with qc=-0.24, <4> !*
23) ! MODULE CAMARASAURIDS with qc=-0.24, <4> !*
24) ! MODULE CETIOSAURIDS with qc=-0.25, <4> !
25) ! MODULE TITANOSAURIDS with qc=-0.27, <4> !*
26) ! MODULE BRACHIOSAURIDS with qc=-0.27, <4> !

27) ! MODULE ASSORTED_SAUROPODS with qc=-0.41, <3> !*

28) ! MODULE THEROPODS with qc=-0.42, <2> !
29) ! MODULE COELUROSAURS with qc=-0.28, <3> !
30) ! MODULE CAENAGNATHIDS_ELMISAURIDS with qc=-0.21, <4> !*
31) ! MODULE OVIRAPTORIDS with qc=-0.21, <4> !*
32) ! MODULE SEGISAURIDS with qc=-0.21, <4> !
33) ! MODULE COMPSOGNATHIDS with qc=-0.21, <4> !
```

```
34) ! MODULE DROMAEOSAURIDS with qc=-0.23, <4> !*
35) ! MODULE GARUDIMIMIDS with qc=-0.23, <4> !*
36) ! MODULE ORNITHOMIMIDS with qc=-0.23, <4> !*
37) ! MODULE SAURORNITHOIDIDS with qc=-0.23, <4> !*
38) ! MODULE ARCHAEOPTERYGIDS with qc=-0.24, <4> !
39) ! MODULE COELOPHYSIDS with qc=-0.24, <4> !
40) ! MODULE COELURIDS with qc=-0.24 <4> !*
41) ! MODULE NOASAURIDS_SHANSHANOSAURIDS with qc=-0.26, <4> !*
42) ! MODULE DEINOCHEIRIDS with qc=-0.26, <4> !*
43) ! MODULE AVIMIMIDS with qc=-0.29, <4> !*

44) ! MODULE CARNOSAURS with qc=-0.32, <3> !
45) ! MODULE SEGNOSAURIDS with qc=-0.25, <4> !*
46) ! MODULE SPINOSAURIDS with qc=-0.26, <4> !*
47) ! MODULE CERATOSAURIDS with qc=-0.26, <4> !
48) ! MODULE DRYPTOSAURIDS with qc=-0.27, <4> !*
49) ! MODULE TYRANNOSAURIDS with qc=-0.27, <4> !*
50) ! MODULE MEGALOSAURIDS with qc=-0.27, <4> !*
51) ! MODULE ALLOSAURIDS with qc=-0.27, <4> !*
52) ! MODULE TERATOSAURIDS with qc=-0.29, <4> !
53) ! MODULE OTHER with qc=-0.32, <4> !
54) ! MODULE THERIZINOSAURIDS with qc=-0.32, <4> !

55) ! MODULE ORNITHOPODS with qc=-0.34, <3> !
56) ! MODULE THESCELOSAURIDS with qc=-0.27, <4> !*
57) ! MODULE PACHYCEPHALOSAURIDS with qc=-0.26, <4> !*
58) ! MODULE HYPSILOPHODONTIDS with qc=-0.28, <4> !*
59) ! MODULE FABROSAURIDS with qc=-0.29, <4> !
60) ! MODULE HADROSAURIDS with qc=-0.29, <4> !
61) ! MODULE HADROSAURINE_DUCKBILLS with qc=-0.27, <5> !*
62) ! MODULE LAMBEOSAURINE_DUCKBILLS with qc=-0.28, <5> !**
----------------------------------------------
R E S U L T :
```

PARASAUROLOPHUS with qc=-0.16

A.3.4 Structure of the taxonomy

If a species is specified during categorization by a number of certain features for each chained category (rule) used to determine it, the *qc* value of the subcategories mostly drops during chaining. This is because the rules in the middle of the taxonomy have the most detailed premises. The premises of the rules in the root of the taxonomy are mostly elementary and are composed of one feature. If this category is present the category has the *qc* value "one". In fig. A.2 the mean *qc* value of twelve hierarchical categorization is shown. The species were described by five features.

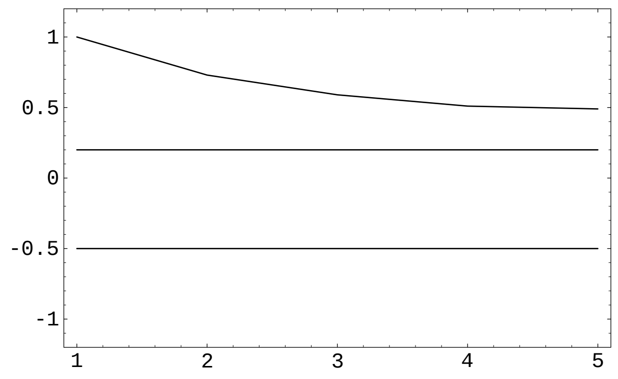


Figure A.2: Mean of qc value during five steps categorization, $\omega = -0.5, \varpi = 0.2$ are also ploted.

Bibliography

[1] M. Abeles. *Corticonics Neural Circuits of the Cerebral Cortex.* Cambridge University Press, 1991.

[2] A. Aertsen, G. Gerstein, M. Habib, and G. Palm. Dynamics of neural firing correlation: Modulation of "effective connectivity". *Journal of Neurophysiology*, 61:900–917, 1989.

[3] J. Aikins. Prototypical knowledge for expert systems. *Artificial Intelligence*, 20:163–210, 1986.

[4] I. Aleksander and H. Morton. *Neurons and Symbols, the Stuff that Mind is made of.* Chapman and Hall, 1993.

[5] A. Allport. Visual attention. In M. Posner, editor, *Foundations of cognitive science*, pages 631–682. The MIT Press, 1989.

[6] J. Anderson. *The Architecture of Cognition.* Harvard University Press, 1983.

[7] J. A. Anderson. *An Introduction to Neural Networks.* The MIT Press, 1995.

[8] J. R. Anderson. *Cognitive Psyhology and its Implications.* W. H. Freeman and Company, fourth edition, 1995.

[9] G. Andreadakis. Neuronale Assoziativspeicher als plausible Zuggeneratoren. Master's thesis, Universität Ulm, Ulm, Germany, 1994.

[10] J. H. Andreae. *Associative Learning: For a Robot Intelligence.* Imperial College Press, London, 1998.

[11] Aristotle. De memoria et reminscentia, aristitle on memory. In J. A. Anderson, editor, *Neurocomputing*, chapter 1, pages 1–10. The MIT Press, 1990. (ca 400 B.C), Richard Sorabji (Trans.).

[12] R. Baeza-Yates and B. Ribeiro-Neto. *Modern Information Retrieval*, chapter 2. Addison-Wesley, 1999.

[13] D. H. Ballard. *An Introduction to Natural Computation.* The MIT Press, 1997.

[14] H. J. Bentz, M. Hagstroem, and G. Palm. Information storage and effective data retrieval in sparse matrices. *Neural Networks*, 2(4):289–293, 1989.

[15] I. Biederman and G. Ju. Surface vs. edge-based determinants of visual recognition. *Cognitive Psychology*, 20:38–6, 1988.

[16] A. Bieszczad. Neurosolver: a neural network based on a cortical column. In *Proceeding of the World Congress on Neural Networks WCNN'94*, pages 756–761, San Diego, CA, 1994. INNS Press, Lawrence Erlbaum Associates.

[17] A. Bieszczad. *Neuromorphic Distributed General Problem Solvers.* PhD thesis, Carleton University, Ottawa, Ontario, Canada, 1996.

[18] S. Biundo. Present-day deductive planning. In *EWSP93*, pages 1–5. IOS Press, 1994.

[19] I. B. Black. *Information in the Brain.* The MIT Press, 1991.

[20] D. S. Blank. *Learning to see Analogies: A Connectionist Exploration.* PhD thesis, Indiana University, Department of Computer Science, Bloomington, Indiana, 1997.

[21] L. Bolc (Ed.), G. Bradshaw, P. Langley, R. Michalski, S. Ohlsson, L. Rendell, H. Simon, and J. Wolf. *Computational Models of Learning.* Springer-Verlag, 1987.

[22] G. Booch. *Object Solutions: Managing the Object-Oriented Project.* Addison-Wesley, 1996.

[23] V. Braitenberg. *Gehirngespinste, Neuroanatomie für kybernetisch Interessierte.* Springer-Verlag, 1973.

[24] V. Braitenberg. Cell assemblies in the cerebral cortex. In R. Heim and G.Palm, editors, *Theoretical Approaches to Complex Systems*, pages 171–188. Springer-Verlag, 1978.

[25] V. Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* The MIT Press, 1984.

[26] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees.* Wadsworth, Belmont, CA, 1984.

[27] L. Brownston, R. Farell, E. Kant, and N. Martin. *Programming Expert Systems in OPS5: An Introduction to Rule-Based Programming.* Addison-Wesley, 1985.

[28] S. Brunak and B. Lautrup. *Neural Networks Computers with Intuition.* World Scientific, 1990.

[29] E. Charniak. A neat theory of marker passing. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, Philadelphia, PA., 1986.

[30] B. Cho, P. Rosenbloom, and C. Dolan. Neuro-soar: A neural-network architecture for goal-oriented behavior. In *Proceedings of Thirteenth Annual Conference of Cognitive Science Society*, Chicago, IL, 1991. Lawrence Erlbaum Associates.

[31] P. M. Churchland. *The Engine of Reason, the Seat of the Soul.* The MIT Press, 1995.

[32] P. S. Churchland and T. J. Sejnowski. *The Computational Brain.* The MIT Press, 1994.

[33] P. R. Cohen. *Empirical Methods for Artificial Intelligence.* The MIT Press, 1995.

[34] A. R. Damasio and H. Damasio. Cortical systems for retrieval of concrete knowledge: The convergence zone farmework. In C. Koch and J. L. Davis, editors, *Large-Scale Neural Theories of the Brain*, chapter 3, pages 61–74. The MIT Press, 1994.

[35] C. Dolan and P. Smolensky. Tensor product production system: a modular architecture and representation. *Connection Science*, 1:53–68, 1989.

[36] G. Dorffner. *Konnektionismus.* B.G. Teubner, Stuttgart, 1991.

[37] C. Downing and S. Oinker. The spatial structure of visual attention. In M. Posner and O. Marin, editors, *Attention and performance XI*, pages 171–187. Erlbaum, Hillsdale, N.J., 1985.

[38] R. O. Duda, J. Gaschnig, and P. E. Hart. Model design in the prospector consultant program for mineral exploration. In D. Michie, editor, *Expert Systems in the Microelectronic Age*. Edinburgh University Press, 1979.

[39] R. Eckhorn, R. Bauer, W. Jordan, M. Brosch, W. Kruse, M. Munk, and H. Reitboeck. Coherent oscillations: A mechanism of feature linking in the visual cortex? *Biological Cybernetics*, 60:121–130, 1988.

[40] J. Elman. Finding structure in time. *Cognitive Science*, 14:179–211, 1990.

[41] A. Elstein, L. Shulman, and S. Sprafka. *Medical Problem Solving - An Analysis of Clinical Reasoning.* Havard University Press, 1978.

[42] J. Feldman. Four frames suffice: A provisional model of vision and space. *Behavioral and Brain Sciences*, 8:265–289, 1985.

[43] J. Ferber. *Les Systèmes Multi-Agents: Versus une intelligence collective.* InterEditions, Paris, 1995.

[44] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving. *Artificial Intelligence*, 2, 1971.

[45] J. Fodor and Z. Pyslyshyn. Connectionism and cognitive architecture: A critical analysis. *Cognition*, 28:121–136, 1988.

[46] C. Forgy. Rete: a fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19:17–37, 1982.

[47] E. Fransén. *Biophysical Simulation of Cortical Associative Memory.* PhD thesis, Stockhokms Universitet, Dept. of Numerical Analysis and Computing Science Royal Institute of Technology, S-100 44 Stockholm, Sweden, 1996.

[48] J. A. Freeman. *Simulating Neural Networks with Mathematica.* Addison-Wesley, 1994.

[49] K. Fukushima. Neocognitron: A hierarchical neural network capable of visual pattern recognition. *Neural Networks*, 1:119–130, 1988.

[50] J. Fuster. *Memory in the Cerebral Cortex.* The MIT Press, 1995.

[51] C. Gary and W. Singer. Stimulus specific neural oscillations in orientation columns of cat visual cortex. *PNAS*, 86:1698–1702, 1989.

[52] G. Gerstein. Interactions within neural assemblies: Theory and experiment. In J.L.McGaugh, N.M.Weinberger, and G.Lynch, editors, *Brain Organization and Memory: Cells, Systems and Circuits.* Oxford University Press, Oxford, 1990.

[53] T. Gilovich. Tversky. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 849–850. The MIT Press, 1999.

[54] R. Givan and T. Dean. Model minimization, regression, and propositional strips planning. In *15th International Joint Conference on Artificial Intelligence*, pages 1163–8, 1997.

[55] K. Gniadek. *Optyczne przetwarzanie informacij.* Wydawnictwo Naukowe PWN, Warszawa, 1992.

[56] R. Goldstone. Similarity. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 763–765. The MIT Press, 1999.

[57] R. Goldstone, D. Medin, and D. Genter. Relational similarity and the nonindipendence of feature in similarity judgments. *Cognitive Psychology*, 23:222–262, 1991.

[58] C. Goller. *A Connectionist Approach for Learning Search-Control Heuristics for Automated Deduction Systems*. PhD thesis, Technical University Munich, Faculty of Computer Science, Munich, 1997.

[59] J. Grainger and A. Jacobs. *Localist connectionist approaches to human cognition*. Erlbaum, 1997.

[60] C. Gross and Mishkin. The neural basis of stimulus equivalence across retinal translation. In S. Harnad, R. Dorty, J. Jaynes, L. Goldstein, and Krauthamer, editors, *Lateralization in the nervous system*. Academic Press, New York, 1977.

[61] U. Hahn and N. Chater. Similarity and rules: distinct? exhaustive? empirically distinguishable? In S. A. Solman and L. J. Rips, editors, *Similarity and Symbols in Human Thinking*, chapter 5. The MIT Press, 1998.

[62] T. Haines. *Walking with Dinosaurs - a Natural History*. BBC Worldwide Limited, 1999.

[63] B. Hammer. On the approximation capability of recurrent neural networks. In *Proc. of the ICSC/IFAC Symp. on Neural Computation*, pages 512–518, 1998.

[64] K. Hammond. *Case-Based Planning: Viewing Planning as a Memory Task*. Academic Press, New York, 1989.

[65] D. Hebb. *The organization of behaviour*. John Wiley, New York, 1949.

[66] D. Hebb. *Textbook of psyhology*. Sanders, Philadelphia London Toronto, 1958.

[67] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, 1989.

[68] W. F. Heiligenberg. *Neural Nets in Electric Fish*. The MIT Press, 1991.

[69] J. Henderson. *Description Based Parsing in a Connectionist Network*. PhD thesis, University of Pennsylvania, Philadelphia, PA, 1994.

[70] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation*. Addison-Wesley, 1991.

[71] D. Hofstadter. *Fluid Concepts and Creative Analogies: Computer Models of the Fundamental Mechanisms of Thought*. Basic Books, 1995.

[72] A. Holst. *The Use of Bayesian Neural Network Model for Classification Tasks.* PhD thesis, Stockhokms Universitet, Dept. of Numerical Analysis and Computing Science Royal Institute of Technology, S-100 44 Stockholm, Sweden, September 1997.

[73] A. Holst and A. Lansner. A flexible and fault tolerant query-reply system based on a bayesian neural network. *International Journal of Neural Systems*, 4(3):257–267, 1993.

[74] J. Hummel. Binding problem. In R. A. Wilson and F. C. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*, pages 85–86. The MIT Press, 1999.

[75] I. Ivanova and M. Kubat. Initialization of neural networks by means of decision trees. *Knowledge Based Systems*, 8:333–344, 1995.

[76] P. Jackson. *Introduction to Expert Systems.* Addison-Wesley, third edition, 1999.

[77] A. Jagota, T. Plate, L. Shastri, and R. Sun. Connectionist symbol processing: Dead or alive? *Neural Computing Surveys, www.icsi.berkeley.edu/ jagota/NCS*, 2:1–40, 1999.

[78] W. James. *Psychology, the Briefer Course.* University of Notre Dame Press, Notre Dame, Indiana, 1985. (Orginally published 1892).

[79] F. Jelinek. *Statistical Methods for Speech Recognition.* The MIT Press, 1998.

[80] M. Jordan. Attractor dynamics and parallelism in a connectionist sequential machine. In *Proceedings of the Eigth Annual Conference on the Cognitive Science Society*, pages 531–546, Hillsdale, 1986. Erlbaum.

[81] J. R. Josephson and S. G. Josephson. *Abductive Inference, Computation, Philosophy, Technology.* Cambridge University Press, 1996.

[82] G. Kahn, A. Kepner, and J. Pepper. Test: a model-driven application shell. In *National Conference on Artificial Intelligence*, pages 814–18, 1987.

[83] J. Kassirer and G. Gorry. Clinical problem solving: A behavioral analysis. *Ann.Intl.Med*, 89:245–255, 1978.

[84] P. Klahr and D. Waterman. *Expert Systems: Techniques, Tools and Applications.* Addison-Wesley, 1986.

[85] T. Kohonen. *Self-Organization and Associative Menory.* Springer-Verlag, 3 edition, 1989.

[86] J. Kolonder. *Case-Based Reasoning.* Morgan Kaufmann, Los Altos, CA, 1993.

[87] B. Kosko. *Neural Networks and Fuzzy Systems: A Dynamical Systems Approach to Machine Intellogence.* Prentice-Hall, 1992.

[88] S. M. Kosslyn. *Image and Brain, The Resolution of the Imagery Debate.* The MIT Press, 1994.

[89] M. Kubat. Second tier for decision trees. In *Proceedings of the 13th Internat. Conference on Machine Learning, ICML'97*, pages 293–301. Morgan Kaufmann, 1996.

[90] M. Kubat, R. Holte, and S. Matwin. Machine learning for the detection of oil spills in satellite images. *Machine Learning*, 30:195–215, 1998.

[91] M. Kubat, I. Koprinska, and G. Pfurtscheller. Learning to classify biomedical signals. In R. Michalski, I. Bratko, and M. Kubat, editors, *Machine Learning and Data Mining: Methods and Applications.* Wiley, 1998.

[92] A. Küchler. *Adaptive Processing of Structural Data: From Sequences to Trees and Beyond.* PhD thesis, University of Ulm, Germany, 1999.

[93] A. Küchler and C. Goller. Inductive learning in symbolic domains using structure-driven recurrent neural networks. In G. Görtz and Hölldobler, editors, *Ki-96: Advances in Artificial Intelligence*, pages 183–197. Springer-Verlag, 1996.

[94] M. Kurbat, E. Smith, and D. Medin. Categorization, typicality, and shape similarity. In *Proceedings of the Cognitive Science Meeting*, Atlanta, GA, 1994.

[95] F. J. Kurfess. Neural networks and structured knowledge. In C. Hermann, F. Reine, and A. Strohmaier, editors, *Knowledge in Neural Networks*, pages 5–22. Logos Verlag, Berlin, 1997.

[96] R. Kurzweil. *The Age of Intelligent Machines.* The MIT Press, 1990.

[97] R. Lacher. Expert networks: Paradigmatic conflict, technological rapprochement. *Minds and Machines*, 3:53–71, 1993.

[98] R. Lacher, S. Hruska, and D. Kuncicky. Backpropagation learning in expert networks. *IEEE Transactions on Neural Networks*, 3:62–72, 1992.

[99] J. F. Laird, A. Newell, and P. S. Rosenbloom. SOAR: An architecture for general intelligence. *Artificial Intelligence*, 40, 1987.

[100] G. Lakeoff. *Women, Fire, and Dangerous Things.* The University of Chicago Press, 1987.

[101] D. Lambert. *Collins Guide to Dinosaurs.* Diagram Visual Information Ltd, 1983.

[102] D. Lambert. *The Ultimate Dinosaur Book.* Dorling Kindersley, 1993.

[103] D. Lancy. *Cross-Cultural Studies in Cognition and Mathematics.* Academic Press, New York, 1983.

[104] T. Lange and M. Dyer. High-level inferencing in a connectionist network. *Connection Science*, 1:181–217, 1989.

[105] T. E. Lange. A structured connectionitst approach to inferecing retrieval. In R. Sun and L. A. Bookman, editors, *Computational Architectures Intergrating Neural and Symbolic Processes: A Perspective on the State of the Artv*, chapter 3, pages 69–115. Kulwer Academic Publishers, 1995.

[106] J. S. Lee, M. Yamaguchi, and N. Ohyama. Integrated image associative memory and its optical implementation. *Optical Review*, 2(4):261–265, 1995.

[107] H. R. Lewis and C. H. Papadimitriou. *Elements of the Theory of Computation.* Prentice-Hall, 1981.

[108] M. S. Livingstone. Kunst, Schein und Wahrnehmung. *Spektrum der Wissenschaft*, 10, 1988.

[109] H. Löllgen. *Management Kardialer Notfälle, American Heart Association.* Ullstein Mosby, Berlin Wiesbaden, 1997.

[110] P. Lucas and L. van der Gaag. *Principles of Expert Systems.* Addison-Wesley, 1991.

[111] G. F. Luger and W. A. Stubblefield. *Artificial Intelligence, Structures and Strategies for Complex Problem Solving.* Addison-Wesley, third edition, 1998.

[112] M. Marcinowski. Codierungsprobleme beim Assoziativen Speichern. Master's thesis, Fakultät für Physik der Eberhard-Karls-Universität Tübingen, 1987.

[113] A. Markov. *A theory of algorithms.* National Academy of Sciences, USSR, 1954.

[114] J. McClelland and A. Kawamoto. Mechanisms of sentence processing: Assigning roles to constituents of sentences. In J. McClelland and D. Rumelhart, editors, *Parallel Distributed Processing*, pages 272–325. The MIT Press, 1986.

[115] J. McClelland and D. Rumelhart. Distributed memory and the representation of general and specific memory. *Journal of Experimental Psychology: General*, 114:159–188, 1985.

[116] J. McClelland, D. E. Rumelhart, and G. Hinton. The appeal of parallel distributed processing. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, chapter 1, pages 3–44. The MIT Press, 1986.

[117] K. A. Meeden. *Towards Planning: Incremental Investigations into Adaptive Robot Control*. PhD thesis, Indiana University, 1994.

[118] D. Merritt. *Building Expert Systems in Prolog*. Springer-Verlag, 1989.

[119] S. Meyers. *Effective C++, 50 Specific Ways to Improve Your Programs and Designs*. Addison-Wesley, 1992.

[120] R. Michalski. How to learn imprecise concepts: A method employing a two-tiered knowledge representation of learning. In *Proceedings of the Fourth International Workshop on Machine Learning*, pages 50–58, Irvine, CA, 1987.

[121] R. Michalski. Two-tiered concept meaning, inferential matching and conceptual cohesiveness. In S. Vosniadu and A. Ortony, editors, *Similarity and Analogy*. Cambridge University Press, 1989.

[122] R. Miikkulainen. *Subsymbolic Natural Language Processing: An Integrated Model of Scripts, Lexicon and Memory*. The MIT Press, 1993.

[123] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, pages 211–77. McGraw-Hill, New York, 1975.

[124] M. Minsky. *The Society of Mind*. Simon and Schuster, New York, 1986.

[125] M. Minsky. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine*, 12(2):52–69, 1991.

[126] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach*. Kulwer Academic Publishers, Boston, 1988.

[127] T. Mitchell. *Machine Learning.* McGraw-Hill, 1997.

[128] T. Mitchell, R. Keller, and S. Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1):47–80, 1986.

[129] M. Moser, O. Ibens, R. Letz, J. Steinbach, C. Goller, J. Schuhmann, and K. Mayr. The model elimination provers SETHEO and E-SETHEO. *Journal of Automated Reasoning*, 18(2), 1997.

[130] G. Murphy and H. Brownell. Category differentiation in object recognition: Typicality constraints on the basic category advantage. *Journal of Experimental Psychology*, 11:70ff, 1985.

[131] J. Neely. Semantic priming effects in visual word recognition: A selective review of current findings and theories. In D. Besner and G. Humphreys, editors, *Basic processes in reading. Visual word recognition.* Erlbaum, 1991.

[132] A. Newell. *Physical symbol sytems.* Norman, 1981.

[133] A. Newell. *Unified Theories of Cognition.* Harvard University Press, 1990.

[134] A. Newell and H. Simon. *Human Problem Solving.* Prentice-Hall, 1972.

[135] A. Newell and H. Simon. Computer science as empirical inquiry: symbols and search. *Communication of the ACM*, 19(3):113–126, 1976.

[136] N. J. Nilsson. *Principles of Artificial Intelligence.* Springer-Verlag, 1982.

[137] N. J. Nilsson. *Artificial Intelligence: A New Synthesis.* Morgan Kaufmann, 1998.

[138] P. Norvig. Marker passing as a weak method for text inferencing. *Cognitive Science*, 13:569–620, 1989.

[139] OFTA. *Les Réseaux de Neurones.* Masson, 1991.

[140] K. Opwis and R. Plötzner. *Kognitive Psychologie mit dem Computer.* Spektrum Akademischer Verlag, Heidelberg Berlin Oxford, 1996.

[141] D. N. Osherson. New axioms for the contrast model of similarity. *Journal of Mathematical Psychology*, 31:93–103, 1987.

[142] D. N. Osherson. Probability judgment. In E. E. Smith and D. N. Osherson, editors, *Thinking*, volume 3, chapter two, pages 35–75. MIT Press, second edition, 1995.

[143] G. Palm. *Neural Assemblies, an Alternative Approach to Artificial Intelligence.* Springer-Verlag, 1982.

[144] G. Palm. Assoziatives Gedächtnis und Gehirntheorie. In *Gehirn und Kognition*, pages 164–174. Spektrum der Wissenschaft, 1990.

[145] G. Palm. Cell assemblies as a guideline for brain research. *Concepts in Neuroscience*, 1(1):133–147, 1990.

[146] G. Palm. Cell assemblies, coherence, and corticohippocampal interplay. *Hippocampus*, 3:219–226, 1993.

[147] G. Palm. The PAN system and the WINA project. In P. Spies, editor, *Euro-Arch'93*, pages 142–156. Springer-Verlag, 1993.

[148] G. Palm and T. Bonhoeffer. Parallel processing for associative and neural networks. *Biological Cybernetics*, 51(201-204), 1984.

[149] G. Palm, F. Schwenker, and A. Bibbig. Theorie Neuronaler Netze 1. Skript zur Vorlesung, 1992. University of Ulm, Department of Neural Information Processing.

[150] G. Palm, F. Schwenker, F. Sommer, and A. Strey. Neural associative memories. In A. Krikelis and C. Weems, editors, *Associative Processing and Processors*, pages 307–325. IEEE Press, 1997.

[151] J. Pearl. *Heuristics: Intelligent Strategies for Computer Problem Solving.* Addison-Wesley, 1984.

[152] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, Palo Alto, CA, 1989.

[153] Y. Peng and J. A. Reggia. *Abductive Inference Models for Diagnostic Problem-Solving.* Symbolic Computation. Springer-Verlag, 1990.

[154] J. Pollack. Recursive distributed representation. *Artificial Intelligence*, 46:77–105, 1990.

[155] M. I. Posner and M. E.Raichle. *Images of Mind.* Scientific American Library, New York, 1994.

[156] M. I. Posner and M. K. Rothbart. Constructing neural theories of mind. In C. Koch and J. L. Davis, editors, *Large-Scale Neural Theories of the Brain*, chapter 9, pages 183–200. MIT Press, 1994.

[157] E. Post. Formal reductions of the general combinatorial problem. *American Journal of Mathematics*, 65:197–268, 1943.

[158] F. Pulvermüller. Syntax und Hirnmechanismen. *Kognitionswissenschaften*, 4:17–31, 1994.

[159] R. Quillian. Semantic memory. In M. Minsky, editor, *Semantic Information Processing*, pages 227–270. MIT Press, 1968.

[160] J. Quinlan. Introduction of decision trees. *Machine Learning*, 1(1):81–106, 1986.

[161] J. Quinlan. *C4.5:Programs for Machine Learning*. Morgan Kaufmann, San Francisco, 1993.

[162] C. Reisbeck. Direct memory access parsing. In J. Koldner and C. Riesbeck, editors, *Experience, Memory, and Reasoning*. Lawrence Erlbaum, Hillsdale, NJ., 1986.

[163] A. Rubin. The role of hypotheses in medical diagnosis. In *Proc. of International Joint Conference on Artificial Intelligence*, pages 856–862. IJCAI, 1975.

[164] D. Rumelhart and McClelland. On learning the past tense of english verbs. In J. McClelland and D. Rumelhart, editors, *Parallel Distributed Processing*, pages 216–271. MIT Press, 1986.

[165] D. Russell and R. Sequin. Reconstruction of the small cretaceous theropod stenonychosaurus inequalis and a hypothetical dinosauroid. *Syllogeous*, 37:1–43, 1982.

[166] S. J. Russell and P. Norvig. *Artificial intelligemce: a modern approach*. Prentice-Hall, 1995.

[167] E. Samkian. VV-XPS, ein Expertensystem zur Ermittlung absetzbarer Werbungskosten und Sonderausgaben bei Einküften aus Vermitung und Verpachtung. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany, 1992.

[168] N. Schwartz. Social cognition: Information accessibility and use in social judgment. In E. E. Smith and D. N. Osherson, editors, *Thinking*, volume 3, chapter ten, pages 345–376. The MIT Press, second edition, 1995.

[169] F. Schwenker. Küntliche Neuronale Netze: Ein Überblick über die theoretischen Grundlagen. In G. Bol, G. Nakhaeizadeh, and K. Vollmer, editors, *Finanzmarktanalyse und -prognose mit innovativen und quantitativen Verfahren*, pages 1–14. Physica-Verlag, 1996.

[170] A. Seitz. Konzeption und Implementierung eines fallbasierten Systems zur Auswertung von multivarianten empirischen Untersuchungen. Master's thesis, University of Ulm, Ulm, 1996.

[171] G. Shafer. *A Mathematical Theory of Evidence.* Princeton University Press, Princeton, NJ, 1976.

[172] L. Shastri. *Semantic Networks: An Evidential Formulation and its Connectionistic Realization.* Morgan Kaufmann, London, 1988.

[173] L. Shastri. Temporal synchrony, dynmic bindings, and SHRUTI- a representational but non-classical model of reflexive reasonning. *Behavioral and Brain Sciences*, 19(2):331–337, 1996.

[174] L. Shastri and V. Ajjanagadde. From simple associations to systematic reasoning: A connectionistic representation of rules, variables and dynamic bindings. *Behavioral and Brain Sciences*, 3(16):417–494, 1993.

[175] E. H. Shortliffe and B. G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975.

[176] H. Siegelmann and E. Sontag. On the computational power of neural nets. *Journal of Computer and System Sciences*, 50:132–150, 1995.

[177] H. A. Simon. *The Science of the Artificial.* The MIT Press, 1969.

[178] H. A. Simon. *Models of my Life.* Basic Books, New York, 1991.

[179] H. A. Simon. Problem solving. In *The MIT Encyclopedia of the Cognitive Sciences*, pages 674–676. The MIT Press, 1999.

[180] W. Singer. Binding by neural synchrony. In R. A. Wilson and F. C. Keil, editors, *The MIT Encyclopedia of the Cognitive Sciences*, pages 81–84. The MIT Press, 1999.

[181] W. Singer and C. Gary. Visual feature integration and the temporal correlation hypothesis. *Ann.Rev.Neuroscience*, 18:555–586, 1995.

[182] D. M. Skapura. *Building Neural Networks.* Addison-Wesley, 1996.

[183] E. Smith, G. Balzano, and J. Walker. Nominal, perceptual, and semantic codes in picture categorization. In *Semantic factors in cognition.* Hillsdale, NJ: Erlbaum Associates, 1978.

[184] E. Smith and S. Sloman. Similarity vs. rule-based categorization. *Memory Cognition*, 22:377–386, 1994.

[185] E. E. Smith. Concepts and categorization. In E. E. Smith and D. N. Osherson, editors, *Thinking*, volume 3, chapter one, pages 3–33. MIT Press, second edition, 1995.

[186] P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations*, pages 194–281. The MIT Press, 1986.

[187] P. Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46:1612–1619, 1990.

[188] S. A. Solman and L. J. Rips. Similarity as an explanatory construct. In S. A. Solman and L. J. Rips, editors, *Similarity and Symbols in Human Thinking*, chapter 1. The MIT Press, 1998.

[189] F. T. Sommer. *Theorie neuronaler Assoziativspeicher*. PhD thesis, Heinrich-Heine-Universität Düsseldorf, Düsseldorf, 1993.

[190] F. T. Sommer and G. Palm. Improved bibdirectional retrieval of sparse patterns stored by hebbian learning. *Neural Networks*, 12:281–297, 1999.

[191] A. Sperduti. Labeling RAAM. *Connection Science*, 6(4), 1994.

[192] L. R. Squire and E. R. Kandel. *Memory. From Mind to Moleculus.* Scientific American Library, 1999.

[193] K. Steinbuch. Die Lernmatrix. *Kybernetik*, 1:36–45, 1961.

[194] K. Steinbuch. *Automat und Mensch.* Springer-Verlag, fourth edition, 1971.

[195] U. Stellmann. *Ähnlichkeitserhaltende Codierung.* PhD thesis, Universität Ulm, Ulm, 1992.

[196] D. G. Stork. Scientist on the set: An interview with Marvin Minsky. In D. G. Stork, editor, *HAL's Legacy 2001's Computer as Dream and Reality*, chapter 2. The MIT Press, 1997.

[197] R. Sun. An introduction: On symbolic processing in neural networks. In R. Sun and L. A. Bookman, editors, *Computational Architectures Intergrating Neural and Symbolic Processes: A Perspective on the State of the Art*, chapter 1, pages 1–18. Kulwer Academic Publishers, 1995.

[198] R. Sun. A two-level hybrid architecture for structuring knowledge for commonsense reasoning. In R. Sun and L. A. Bookman, editors, *Computational Architectures Integrating Neural and Symbolic Processing*, chapter 8, pages 247–182. Kluwer Academic Publishers, 1995.

[199] R. Sun. Learning, action, and consciousness: a hybrid approach towards modeling consciousness. *Neural Networks*, 10(7):1317–1331, 1997.

[200] R. Sun and T. Peterson. Some experiments with a hybrid model for learning sequential decision making. *Information Sciences*, 111:83–107, 1998.

[201] G. Sussman. *A Computer Model of Skill Acquisition.* MIT, Cambridge, 1975.

[202] R. Sutton and A. Barto. Time-derative models of pavlovian reinforcment. In M. Gabriel and J. Moore, editors, *Learning and Computational Neuroscience: Foundations of Adaptive Networks.* The MIT Press, 1990.

[203] R. Sutton and A. Barto. *Reinforcment Learning. An Introduction.* The MIT Press, 1998.

[204] A. Tarski. The semantic conception of truth and foundations of semantics. *Philos. and Phenom. Res.*, 4:241–376, 1944.

[205] A. Tarski. *Logic, Semantics,Metamathematics.* Oxford University Press, London, 1956.

[206] A. Tarski. *Pisma logiczno-filozoficzne. Prawda*, volume 1. Wydawnictwo Naukowe PWN, Warszawa, 1995.

[207] H.-H. Teh. *Neural Logic Networks: A New Class of Neural Networks.* World Scientific, 1995.

[208] G. Touretzky, D.S. amd Hinton. Symbols among the neurons: Detals of a connectionist inference architecture. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 238–243, San Mateo, CA, 1985. Morgan Kaufman.

[209] G. Touretzky, D.S. amd Hinton. A distributed connectionist production system. *Cognitive Science*, 12:423–466, 1988.

[210] A. Triesmann and S. Gormican. Feature analysis in early vision. *Psyhology Review*, 95:15–30, 1988.

[211] A. M. Turing. On computable numbers, with an aoolication to the entscheidungsproblem. *Proc.London.Math.Soc.*, 42(2):230–265, 1936.

[212] A. Tversky. Feature of similarity. *Psychological Review*, 84:327–352, 1977.

[213] A. Tversky and Kahneman. Availability: A heuristic for judginig frequency and probability. *Cognitive Psychology*, 5:207–232, 1973.

[214] C.-H. Tzeng. *A Theory of Information in Game-Tree Search.* Springer-Verlag, 1988.

[215] J. van Hemmen. Hebbian learning and unlearning. In W. Theumann, editor, *Neural Networks and Spin Glasses*. World Scientific, 1990.

[216] H. Vollbrecht. Three principles of hierarchical task composition in reinforcement learning. In *ICANN98*, pages 1121–1126. Springer-Verlag, 1998.

[217] I. Watson and F. Marir. Case-based reasoning: A review. *The Knowledge Egineering Review*, 9(4):355–391, 1994.

[218] D. Weld. An introduction to least-commintment planning. *AI Magazine*, 15(4):27–61, 1994.

[219] T. Wennekers. *Synchronisation und Assoziation in Neuronalen Netzen*. Shaker Verlag, Aachen, 1999.

[220] S. Wermter. *Hybrid Connectionist Natural Language Processing*. Chapman and Hall, London, 1995.

[221] A. Wichert. *Associative Class Library and its Applications*. Department of Neural Information Processing , Universität Ulm, Ulm, Germany, 1988.

[222] A. Wichert. MTCn-nets. In *Proceedings World Congres on Neural Networks*, pages 59–62. Lawrence Erlbaum, 1993.

[223] A. Wichert. Hierarchical categorization. In M. W. Evens, editor, *Ninth Midwest Artificial Intelligence and Cognitive Science Conference*, pages 141–148. AAAI Press, 1998.

[224] A. Wichert and H. A. Kestler. A categorical based reanimation expert system. In E. Ifeachor, A. Sperduti, and A. Starita, editors, *Neural Networks and Expert Systems in Medicine and Healthcare*. World Scientific, 1998.

[225] W. A. Wickelgren. Context-sensitive coding, associative memory, and serial order in (speech)behavior. *Psychological Review*, 76:1–15, 1969.

[226] W. A. Wickelgren. *Cognitive Psychology*. Prentice-Hall, 1977.

[227] D. E. Wilkins. "that's something I could not allow to happen": HAL and planning. In D. G. Strok, editor, *HAL's Legacy 2001's Computer as Dream and Reality*, chapter 14, pages 305–331. The MIT Press, 1997.

[228] D. Willshaw, O. Buneman, and H. Longuet-Higgins. Nonholgraphic associative memory. *Nature*, 222:960–962, 1969.

[229] P. H. Winston. *Artificial Intelligence*. Addison-Wesley, third edition, 1992.

[230] Q. Yang. *Intelligent Planning A Decomposition and Abstraction Based Approach*. Springer-Verlag, 1997.

[231] L. A. Zadeh. Fuzzy logic and approximate reasoning. *Synthese*, 30:407–428, 1975.