Testing mechanisms of development within a computational framework

Gary Jones
Psychology Department
University of Nottingham
Nottingham NG7 2RD
England

Thesis submitted to the University of Nottingham for the degree of Doctor of Philosophy, November, 1998.

Abstract

Theories of development have proposed several mechanisms by which development occurs in children. The majority of the proposed mechanisms lack precise definitions, and are difficult to test individually whilst holding the effects of all other mechanisms constant. Implementing the mechanisms within a computational framework forces precision and enables the effect of each mechanism to be examined in isolation. A computational model of adult behaviour in a developmental task was created. The model included a range of the mechanisms proposed by theories of development, whereas previous computational models of development have examined very few mechanisms. The mechanisms were tested in the model both independently and in combination, with the results being compared against the behaviour of seven year old children on the task. The independent modifications showed that the behaviour of the model changes significantly for four mechanisms: strategy choice, strategy accuracy, capacity, and processing speed. The best mechanism (strategy accuracy), when applied to the adult model, matched seven out of nine regularities in the behaviour of seven year olds, including reaction time and errors. The combined modifications also matched seven year old children's behaviour. The results show that a range of developmental mechanisms can now be routinely tested and evaluated within a single computational model. The method of modifying computational models is an interesting way to examine the influences of developmental mechanisms, and therefore helps in answering "What develops in children?".

1. Introduction

> "If we can construct an information processing system with rules of behavior that lead it to behave like the dynamic system we are trying to describe, then this system is a theory of the child at one stage of development. Having described a particular stage by a program, we would then face the task of discovering what additional information processing mechanisms are needed to simulate developmental change – the transition from one stage to the next. [...] Thus, the theory would have two parts – a program to describe performance at a particular stage and a learning program governing the transitions from stage to stage." Herbert A. Simon, 1962, pp154-155.

Simon's quote lays out a field of work that is still in its infancy. This idea is important for two reasons. First, it implies that understanding child development is concerned with two elements: the characterisation of behaviour at a particular time and how behaviour changes across time. Second, it puts forward the idea that information processing, specifically computational modelling (the contemporary term for a program in this domain), can help in understanding the processes behind the two elements of child development. This thesis uses computational modelling to examine child development, and puts forward a new methodology by which this examination can take place. The conclusion is that computational modelling under this new methodology is a fruitful way to explore the processes that underlie development.

This chapter outlines the problems with current empirical methods of studying child development. Based on these problems, it is argued that an additional non-empirical method, that of computational modelling, can help in examining child development. An overview of current computational models of development and the environments in which they are written is given. This highlights cognitive architectures as an area in which the proposed mechanisms influencing child development can be examined. Cognitive architectures have often been ignored in models of development. The chapter concludes by putting forward a methodology of modifying cognitive architectures as a way in which child development can be studied.

1.1 A précis of child development

In general, as children get older they become more adept at solving tasks, and can solve tasks which they could not do before. Early research in child development proposed that this occurs because children go through a chronological sequence of stages, whereby their

behaviour at any one time is governed by what stage of development they are in (e.g. Piaget, 1952). This conclusion implies a discontinuous view of development. Performance should be relatively stable until the child reaches the end of a stage and begins the next stage, whereupon a leap in performance should be seen.

Viewing development as a sequence of stages leads to the prediction that tasks using the same reasoning should be mastered at the same time, or at least within the same stage. For some problems, however, this is not the case. The concept of conservation, for example, is mastered at different times for different conservation tasks (e.g. Elkind, 1961). Tasks which use the same reasoning that are mastered at different developmental times present a more continuous view of development (see Rosser, 1994). Theories of development have had to accommodate both views because whilst gradual changes do occur in children's behaviour, children are also able to show abrupt changes in behaviour in short spaces of time.

1.2 Methods of studying child development empirically

The majority of child development studies have concentrated on characterising children's behaviour at different ages, or levels of performance (usually on a specific task); only a minority have studied how children's performance improves. Two problems exist with studying development in a stage based way. First, it impedes the understanding of how children move from one level of performance to another (Siegler & Shipley, 1995). Second, it creates an impression of children's thinking as conforming to one state at one time and another state at another time (Siegler, 1995).

Examining how children's performance improves, and detailing what changes occur during development, is difficult. This is mainly because development occurs over a number of years. Observations spanning several years are necessarily sparse which means that the mechanisms (defined as "any mental process that improves children's ability to process information", Siegler, 1989, p.354) that give rise to the changes in performance have to be inferred (Siegler, 1995).

Siegler and Jenkins (1989) use the microgenetic approach to examine changes in performance. This involves observing behaviour as much as possible during the onset of the new behaviour, until the new behaviour becomes stable. The approach is fruitful but the time span of the change in behaviour must realistically be limited to a number of

weeks. The applicability of the microgenetic approach to the study of development in general may be limited.

A dilemma is now posed: it is important to understand what changes during development, but it is very difficult to examine such changes empirically. Studies involve either examining development on a specific task, where the change occurs over a short period of time, or examining general development, over a number of years, but observing the behaviour of the children relatively few times over that period. The former leads to problems of generalising any conclusions to development as a whole; the latter leads to problems of missing important aspects of development because the time span between observations is too large (Newell, 1972, hints at this problem).

The problems of studying development that have been outlined probably contribute to the reason why there are so many theories of development. Inferring the mechanisms involved is difficult because of the time span of observations. Each theory of development is difficult to test empirically because controlling for factors like knowledge and processing capacity is very hard to do. A method for examining development which is outside the scope of empirical study must be considered.

1.3 How computational models can help examine child development

Computational modelling involves characterising human behaviour in information processing terms. Taken to its extreme, this means characterising behaviour by means of a computer program. Defining behaviour on a task using a computational model can help to characterise how the different behaviours seen at each individual level of task performance are generated. This enables a fine-grain account of the processes that are involved in producing the task behaviour (Simon & Halford, 1995).

A model that matches the observed behaviour at one level of task performance can suggest what knowledge and procedures children may be using to generate that level of task performance. To the extent that behaviour on the task cannot be routinely measured, the model can make predictions about the missing elements. Computational modelling also allows manipulations that cannot be performed on subjects, such as changing specific pieces of knowledge (e.g. Gentner, Rattermann, Markman & Kotovsky, 1995; McCloskey, 1991). The manipulations that can be made to a computational model can be made in isolation, which allows a single manipulation (e.g. knowledge) to be made whilst controlling for all other factors.

In order to move from one level of task performance to another, or improve behaviour at the same level of task performance, the model must have mechanisms which enable a transition in the model's behaviour to occur. The transition mechanisms will put forward ways in which development takes place. The transition mechanism can itself be viewed as a theory of development.

The benefits that computational modelling can give to the study of development require detailed empirical studies to have taken place. Detailed descriptions of the behaviour at each level of task performance are required to build and test the behaviour of the model. The development of the model therefore becomes the development of a theory of how the task is performed. Ideally the model should be able to make predictions regarding not only performance on the specific task but also performance on other tasks. This will show the generality of the mechanisms used in the model.

Models of development have succeeded in both modelling behaviour at different levels of performance (e.g. Klahr & Siegler, 1978), and to a limited extent the transition across performance levels within the same model (e.g. Jones & VanLehn, 1991; McClelland & Jenkins, 1991). Further models have been able to simulate the transition across performance levels, or performance on other similar tasks, by the addition of knowledge (e.g. Klahr & Wallace, 1976; Young, 1973).

All models of development have been able to provide a close match to at least some of the behaviour of children at different levels of performance. In this way they are able to put forward the processes that the child may be using when performing specific tasks. Less success has been achieved in getting the same model to transcend different levels of performance.

Jones and VanLehn (1991) examine the transition from the sum to the min strategy in childrens' simple addition. Their model learns strategies by constantly striving to perform the task more efficiently. The model omits some strategies (such as retrieval), and does not provide a match to the children's data. McClelland and Jenkins (1991) model strategy acquisition in the balance-beam task. The model learns from experience of success and failure on the task, but includes contentious architectural assumptions and requires much more specific task experience than children ever need. Both models view transition as changes in knowledge.

The models which implement a transition mechanism show that modelling transition is very difficult. There is not only a general agreement on this point (e.g. Newell, 1990), but also that "one of the key unsolved problems in cognitive development is the precise specification of developmental transition mechanisms" (Shultz, Schmidt, Buckingham & Mareschal, 1995, p.205).

1.4 A new method for using computational modelling to study child development

Current computational attempts at providing a satisfactory transition mechanism have only accounted for knowledge as a mechanism of development. Theories of development propose a variety of mechanisms by which development occurs. Development is said to involve changes in knowledge (e.g. Piaget, 1952; Vygotsky, 1978), changes in strategy choice (Siegler & Shrager, 1984), changes in processing capacity (e.g. Case, 1985; Pascual-Leone, 1969), changes in processing speed (Kail, 1986), and so on.

Ignoring the other mechanisms of development within a computational model may contribute to the difficulty of implementing a transition mechanism, because transition is restricted to a single route: knowledge. A way forward in examining transition mechanisms could be to test the mechanisms of development that are proposed by developmental theories. Examining the influence of all proposed mechanisms of development will provide a focus for what transition mechanisms must accommodate.

Computational modelling can help examine the influence of different developmental mechanisms. A computational model of a developmental task can be created which incorporates all proposed mechanisms of development. This would allow one mechanism to be manipulated whilst all others are controlled for. The influence of each developmental mechanism can be examined, because any change in the model's behaviour will be solely due to the manipulated mechanism.

The method of modifying computational models to examine the influence of developmental mechanisms can therefore help examine transition mechanisms. First, it allows the testing of mechanisms of development that are proposed by developmental theory. Second, it provides a focus for transition mechanisms because it elicits the mechanisms that influence behaviour from those that do not.

1.5 The use of cognitive architectures as a modelling format

There now exist cognitive architectures which support the development of computational models within a theoretical framework. The architectures usually incorporate some general psychological mechanisms, including those proposed by developmental researchers (e.g. the representation and characteristics of working memory) which constrain the way task behaviour can be modelled (for discussion, see Newell, 1990). The goal of cognitive architectures is to explain performance on a wide range of tasks within a single theoretical framework (e.g. Kail & Bisanz, 1992).

The inclusion in these architectures of proposed mechanisms by which development may occur is an advantage which is yet to be explored. Whilst most current models of development have been successful in being able to produce behaviour at different levels of performance by only adjusting knowledge, this does not reveal the extent which other mechanisms may influence development. Cognitive architectures can explore these influences.

The use of architectures also provides other benefits. In most cases, developmental models *have not been created within an architecture*. This is partly because the models were created before such architectures existed, and partly because it was not their goal to examine the influences of architectural mechanisms on development. The models are therefore less constrained than ones written within architectures. Modellers can manipulate the computational model in an unconstrained way so that it is able to fit the subject data. The modeller can also use mechanisms which are outside of those proposed by theories of development, some of which may not be psychologically plausible.

A lack of constraints can leave models open to the above criticisms because they may use any mechanisms in order to match the subject data. Modelling within architectures should reduce the scope of criticism. In particular, the architectures include timing estimates (of processing speed), which enable a direct matching of the temporal behaviour of the model with that of subjects.

Timing estimates provide a major constraint in how a task is modelled (particularly a sequential task), because timed behavioural components of subjects should take the same amount of time for the model. When this is not the case, it indicates areas where the model must be improved. Such precise predictions as reaction times have not been

provided by any models of development because they have failed to include a timing component.

The use of architectures therefore provides a more rigorous basis for testing the behaviour of the model. Architectures provide a more constrained modelling environment, and provide a way of exploring the influences that various developmental mechanisms (proposed by developmental psychologists) have upon behaviour.

1.6 Chapter summary and overview of thesis

This thesis proposes a new approach to examining development computationally. Current models of development have achieved success in modelling single performance levels, but have not achieved a similar degree of success in modelling the transition mechanisms that allow changes in performance. The models which implement transition mechanisms have concentrated on knowledge as the sole mechanism of development. Knowledge is only a subset of the mechanisms of development that are proposed by theories of child development.

The new approach in studying development computationally will modify a model of a developmental task, implemented in a cognitive architecture, in a variety of ways motivated by different theories of development. Each developmental mechanism that is proposed by developmental theories will be examined with respect to the model to see what influence it has on the model's behaviour. The model will be able to test and evaluate theories of what develops.

To illustrate this approach, a model of adult behaviour on a developmental task will be modified in independent ways to see the effect that each mechanism of development has upon task performance. The behaviour of the modified model will be compared to the behaviour of seven year old children on a developmental task. Selected modifications will then be used together to examine if combinations of modifications match subject data better than the independent modifications. The results will provide a comprehensive guide as to which modifications have the most influence on task performance, and therefore which mechanisms should be the focus of attention for researchers developing transition mechanisms.

A review of the developmental theory literature will be given, which suggests various mechanisms by which development might occur. Computational models of development

are then discussed, highlighting the lack of coverage of many proposed mechanisms of development. A sample task in which to study development is then presented, and the results from studies involving both adults and children on the task are described. A consideration to the possible architectures in which this task could be modelled is given, followed by an in-depth description of the model of the adult behaviour on the task. This will present all of the possible modifications that can be made to the model.

The model is then modified in several independent ways, with the resulting behaviour from each modification being compared to that of seven year old children on the task. The theoretical motivation behind each modification is described before each comparison with data. The effect of interacting modifications is then examined. The thesis concludes with a discussion of the contributions that the methodology of modifying architectures gives to studying child development.

2. Theories of development

This chapter briefly summarises the three most influential theoretical approaches to child development: Connectionist, Piagetian, and Information-Processing. The Piagetian and Information-Processing theories form the basis of many theories of child development, which will be discussed in less detail. Each theory is categorised in terms of the mechanisms by which it explains how development occurs. This provides a list of hypothesised developmental mechanisms by which change may occur, which are described in tabular form. The table will be used in later chapters when assessing how many of the mechanisms have been implemented within computational models, and when testing the mechanisms within a computational model. The chapter ends by highlighting two problems of theories of development: they lack precision and are difficult to test. The two problems are ones which computational modelling can help overcome.

2.1 Introduction

There are now many theories of child development. Three general theories stand out: Connectionist, Piagetian, and Information-Processing. Most other theories have been developed recently, and often mix aspects of Piagetian and Information-Processing theories (Flavell, 1984). For this reason the theory formed by Piaget, and the Information-Processing theory, are explained in detail, whereas connectionist theory is only covered briefly. Describing the Piagetian and Information-Processing theories will help in understanding some of the other theories of development. A critique of all three theories is only provided when the criticism is rectified by one of the later theories.

One further reason for covering Information-Processing theories in depth is that in their most extreme form, the theory is implemented as a computer program, or computational model (Klahr, 1992). This obviously has direct relevance to examining development via computational modelling.

Within each developmental theory, knowledge structures tend to be labelled using different terms (such as Piaget's schemas). The generic term "knowledge structure" will therefore be used throughout this chapter to refer to any representation of knowledge within a developmental theory, unless the theory is described in enough detail to differentiate types of knowledge.

2.2 Connectionist theory

Connectionist theory[1] is a computational framework which is intended to be an approximation to brain function (e.g. Elman, Bates, Johnson, Karmiloff-Smith, Parisi & Plunkett, 1996). The framework is a network consisting of a set of input units and a set of output units (and possibly other intermediate sets of units, such as context or hidden units) which are linked by weights. The units can be thought of as neurons, and the weights as the links between neurons. Each unit has an activation, and a combination of this and the size of the weights between one unit and all of its linked units determines the effect the unit has on its linked neighbours (e.g. a negative weight indicates an inhibitory link, a positive weight indicates an excitatory link). Both activation and weight are expressed as real numbers, usually ranging between the values of -1 and +1. A learning mechanism is applied during the training of the network, whose job it is to alter the weights of the network so that there is a minimal error between the network's output and the desired output to a given input.

A network usually begins with a pre-determined number of input, output, and intermediate units, and will have a random set of weights. Training involves the network seeing a specific input and trying to match it to a specific output, using the learning mechanism to minimise the error between the two. As activation and weight are real numbers, it is always likely that the networks output will not match the desired output for some time, and so learning is a gradual process.

Connectionist theory proposes that development is ongoing and occurs gradually. Although stage behaviour may manifest itself in the output of a connectionist network, the network is in fact learning gradually with every input it sees. The mechanism enabling development is a combination of the type and variation of input the network sees, and the learning mechanism that is applied to the network. The developmental progression in the behaviour of the network can be seen by testing the network after different amounts of training.

Connectionist networks have received some criticism (e.g. Fodor & Pylyshyn, 1988), although they have been applied successfully to a wide range of developmental domains

---

[1]For a full description of connectionist (or neural) networks, the reader is referred to either Rumelhart, McClelland, and the PDP research group (1986) or Bechtel and Abrahamsen (1991).

such as object permanence (Mareschal, Plunkett & Harris, 1995), morphology (Plunkett & Marchman, 1993), the balance-beam problem (McClelland & Jenkins, 1991; Shultz, Mareschal & Schmidt, 1994), concepts of distance, time, and velocity (Buckingham & Shultz, 1994), and seriation (Shultz, Schmidt, Buckingham & Mareschal, 1995). Some of these connectionist networks will be covered in Chapter 3.

2.3 Piagetian theory

The central theme of Piagetian theory (Inhelder & Piaget, 1958; Piaget, 1950; Piaget, 1952) is a characterisation of children's behaviour at different ages, based on a concept of stages. Four stages exist, which are sequential, such that a child cannot progress to a particular stage until they have been through the stages that precede it.

In the sensori-motor stage (from birth to eighteen months) the central behaviour of the child concerns interactions with the external world. These interactions enable the child to develop simple behavioural knowledge structures (e.g. lifting an arm) which eventually become co-ordinated (e.g. lifting an arm, moving it to an object, grasping the object). Children in this stage are egocentric; they are unable to make any distinction between themselves and the rest of the world (Donaldson, 1978). The child's egocentricity also influences its interactions with the environment; children in this stage have to make actions on the environment, because they cannot form their own internal representations of that environment.

The pre-operational stage (from eighteen months to seven years) has the main achievement of forming internal representations of the external world. Objects no longer need to be present in order for children to think about them. This leads to the development of operations towards the end of this stage. An operation can be thought of as an action performed on an internal representation. That is, it occurs in the mind rather than having to occur in the environment itself. The pre-operational stage bears great resemblance to the sensori-motor stage. In the sensori-motor stage, it was the physical environment that was acted upon to develop action knowledge structures. In the pre-operational stage, it is the internal representations that are acted upon to form operations.

The concrete operational stage (from seven to eleven years) sees the operations become more consolidated and general. The child is able to co-ordinate two mentally represented events rather than paying attention to only one at a time. This results in being able to notice relationships between knowledge structures, yielding mental operations.

In the formal operational stage (from eleven years onwards), children take on a more abstract and scientific viewpoint. They are able to create and test hypotheses about things, and integrate the results of these into general principles (by co-ordinating mental operations to create formal operations).

Progression through the stages of development occurs by the creation and elaboration of knowledge structures (eventually being integrated into larger systems, Beilin, 1992). The child begins with an innate set of knowledge structures which are built upon throughout development. Piaget believed that the form of children's knowledge could be described by means of symbolic logic (Case, 1985), thus knowledge structures are often referred to as logical structures.

The processes which provide the impetus for development are equilibrium and disequilibrium. These are self-regulatory processes which constantly strive to achieve a state of mental stability (when all of the child's knowledge structures fit in with the events and perceptions of the external world). When this is not the case, the processes of accommodation and assimilation must take place in order for the mind to return to a state of stability.

Accommodation involves changing knowledge structures to fit in with new information about the world; assimilation involves changing new information about the world so that it fits in with existing knowledge structures. Accommodation can be seen as catering for the growth and change in knowledge structures, and assimilation as the preservation of knowledge structures (Donaldson, 1978). The two processes never exist in isolation; even in extreme cases, there will be elements of each process (Siegler, 1991, p.23).

The idea of stages suggests that development is discontinuous, because behaviour in each stage is qualitatively different from any other stage. However, Piaget insists that development is continuous, because structures are constantly being updated and elaborated throughout each stage. The continuous/discontinuous view is held to be a problem of scale, where a behaviour can appear continuous at one level of granularity yet discontinuous at another (Piaget, 1960).

Development in Piagetian theory is a slow process because accommodation and assimilation both require pre-existing knowledge structures in which to integrate new

information. Where this is not the case, the child cannot profit from its experiences with the external world (Case, 1985). This is part of the reason why the child must go through the described stages in sequence. Each subsequent stage builds upon the knowledge structures of the previous stage. For example, the pre-operational stage cannot internalise physical actions into operations before those physical actions have been developed as sensory-motor structures. A child that is characterised as being in one stage should not, therefore, be able to exhibit behaviour which requires basic concepts from another, advanced, stage.

The methods of cognitive change (the processes of accommodation, and assimilation, when in disequilibrium) have been criticised as being vague (see Flavell, 1996), and lacking a precise definition of when the processes occur (Wohlwill, 1966). The logical form of the knowledge structures suggests that tasks which involve the same underlying logical concepts should all be solved as soon as the particular logical concept is acquired. This is often not the case (e.g. Gelman, 1972), a problem Piaget referred to as *horizontal décalage*. The exact methods by which accommodation and assimilation change the knowledge is not discussed. The general lack of specificity in explaining cognitive change means it is easy to interpret a wide range of results as being compatible with the theory (Klahr & Wallace, 1976). The theory is also criticised regarding the nature of stage behaviour (e.g. Elkind, 1961). These criticisms will not be discussed here, although later in this chapter several theories based on Piaget's are described, which attempt to remove the weaknesses of the theory.

2.4 Information-Processing theory

Information processing theory is not a single, general, theory of development like that of Piaget. It is really a framework or language from which individual information processing theories are developed, and as such there is no single instantiation of the theory that is generally agreed upon. However, any information processing theory must meet a set of assumptions, the central one being that cognitive behaviour can be described in terms of processes that manipulate knowledge (e.g. Klahr, 1992). Children's thinking is therefore assumed to be characterised by what information is processed, the capacities that limit how much information is processed, and the methods by which the information is processed (Siegler, 1991). Rather than looking at developmental stages, the theory considers children's thinking at a particular age to depend upon how well they can represent, transform, and process information.

The information processing system of the child is also assumed to be self-modifying. This means that whilst the environment can influence development, it is ultimately up to the internal mechanisms to modify processes and knowledge into the more advanced forms that are evidenced during development.

The basic assumptions that have been described for information-processing theory assert two distinct types of mechanism; those for storage of information, and those for processing information. Figure 2.1 shows how these mechanisms may interact with one another. The figure is not based on any specific theory, although the mechanisms are generally agreed upon by researchers when describing information processing theory (e.g. Siegler, 1991; Simon, 1972). Three types of store exist: sensory memory, working memory, and long-term memory. The memories are distinct because each has different properties.

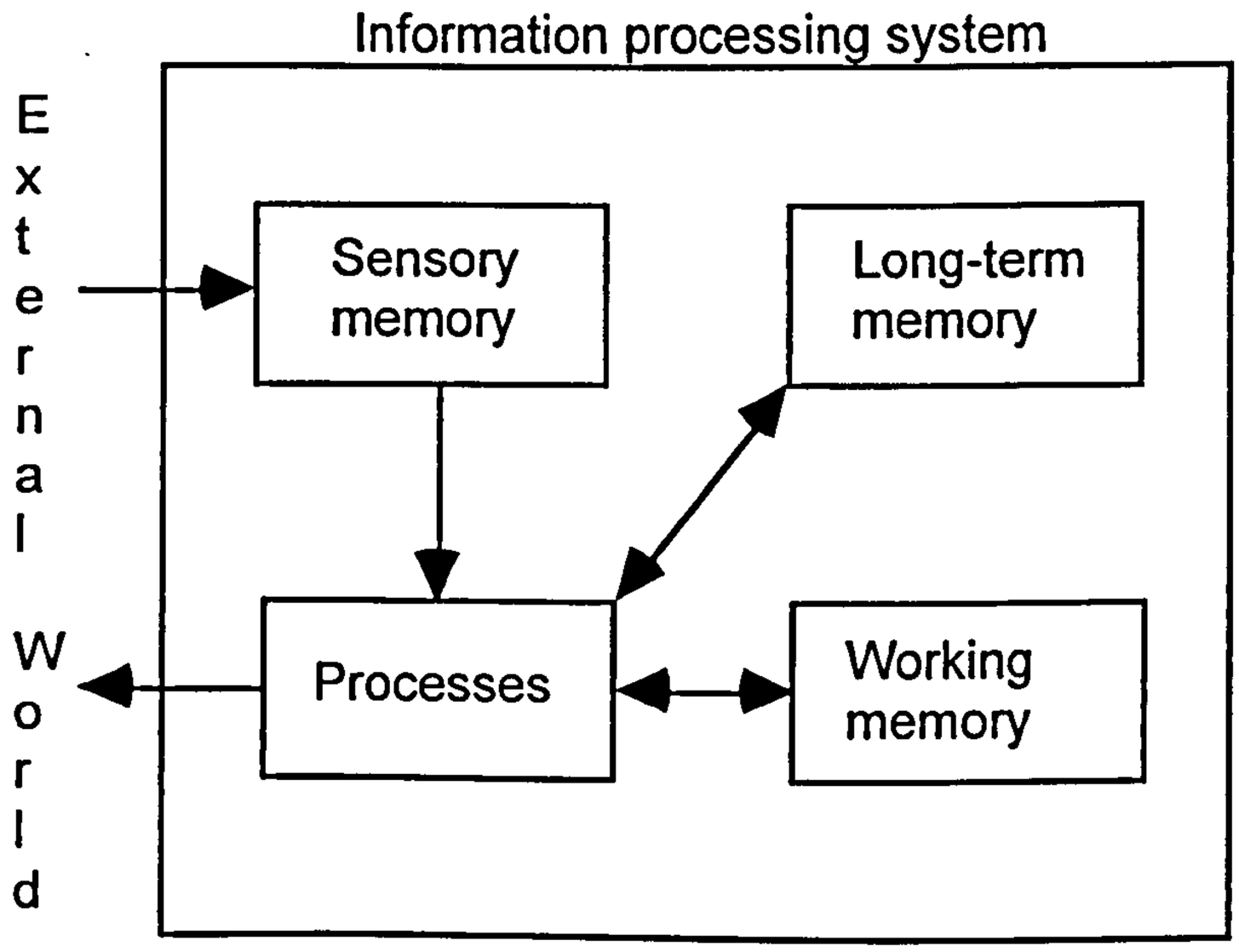


Figure 2.1: The flow of information between the stores and processes involved in the information processing system.

Sensory memory is a buffer between the external world and the mind's architecture. All information from the external world that is obtained from the sensory organs enters sensory memory. This information is very short-term (typically up to one second for visual information, Simon, 1972). Due to the amount of sensory information available, and its highly short-term nature, only a subset of the information is passed through to working memory. Various processes exist which must decide what sensory information is

important and therefore what information in sensory memory is placed in working memory.

Working memory is a temporary store that processes can use to place elements from sensory and long-term memory. The information held in working memory is usually of a declarative nature. The store allows rapid retrieval, but has both a limited capacity and a limited time span (although these limitations can be overcome by using strategies; for example, chunking can improve capacity, and rehearsal can improve time span). Working memory is the only store that allows conscious access, and therefore it is the only memory information that is reportable from introspection.

Long-term memory is unlimited in capacity and in duration. Information in long-term memory is stored in one of two general formats: declarative (facts), and procedural (skills and strategies). There is no direct link between long-term memory and working memory because some processing must occur to combine the two. For example, the process of recalling the name of a person will involve retrieving the features of the person's face from working memory, and using these to examine long-term memory to retrieve the name of the person. Similarly, remembering something that is in working memory (and hence placing it in long-term memory) can be done using a process of rehearsal.

There are four main processes which facilitate the manipulation of information (Siegler, 1991). Each of these plays an important role in how information processing theory accounts for development:

- Automatisation: When first performing a task, it has to be consciously attended to in order to avoid making mistakes. With enough practice, the task becomes automatic and requires relatively little attention. The role of automatisation therefore frees cognitive resources so that further processes can occur whilst performing the automated task.
- Encoding: There is too much sensory information to be encoded into working memory in one pass. Encoding processes must decide what sensory information is appropriate, and how to encode this information. How a task is encoded therefore has a direct influence on performance of the task. Failure to encode the most relevant task features, or failure to encode them efficiently, can therefore intrude upon learning.
- Generalisation: Finding commonalities between different concepts helps in understanding them, and it provides a method of proceeding in novel tasks which

share concepts with known tasks. Encoding plays a major role in finding commonalities because concepts must be represented in similar ways.

- Strategy acquisition: Through an interaction of one or more of the above, new strategies for doing tasks may emerge.

Piagetian theory was criticised for being too vague. Information processing theories range from being vague and open-ended (a verbal theory which does not tie itself to the particular methods by which, for example, long-term memory will be represented) to being very specific (implementing the theory as a computer program which, by necessity, ties down every aspect of the theory to specific representations and processes). The latter has the advantage of removing any theoretical ambiguities (Klahr, 1992).

Other theories of development, some of which integrate Piagetian and Information processing perspectives (so-called neo-Piagetian theories, Beilin, 1994), are now reviewed. The theories range from being implemented as verbal theories, to formal theories, which involve some form of computer model. Examining each theory will result in the formation of a table of proposed mechanisms by which development occurs.

2.5 Other theories of child development

Rather than listing and detailing further theories of development in turn, each theory is categorised by its primary explanation of how development occurs (knowledge based, socially based, capacity based, processing based), and is explained in brief detail. This should enable the similarities between each theory within the same category to become apparent.

2.5.1 Knowledge based theories

Although every developmental theory puts forward knowledge as a major influence on development, two theories stand out as being purely knowledge based, those of Klahr and Wallace, and Siegler. These will be detailed in turn.

2.5.1.1 Klahr and Wallace

Klahr and Wallace (1976) detail an information-processing theory of development, the central theme of which is a timeline, which is a record of all events that have taken place in the information processing system. This assumes that children store a record of all events they have accrued during the day, including records of goal creation and

completion, and all behavioural activities. The theory is based on a three tier hierarchy of production systems, each beginning with an innate kernel of productions.

The productions in the first tier form the basis of behaviour. Initially the first tier consists of productions which receive incoming stimulus from the external world. New productions in the first tier are created based on examination of the timeline (i.e. they are created from experience with the external world). This enables the first tier productions to eventually be able to elicit a whole range of behaviour. The second tier productions are for general problem solving methods, and are used when no productions in the first tier are appropriate. This means that behaviour will tend to be based on previous experience, because tier-one productions are used whenever possible. The third tier contains the productions that examine the timeline and create new tier-one productions.

The timeline is critical because it is used to create productions in the first tier. Productions can be created using both the timeline and existing productions. New productions which are based on existing productions will either be more general than the previous production (by detection of regularities), or more specific than the previous production (by detection of redundant information).

Development occurs in a continuous fashion, because productions are continually being created which make behaviour more efficient and adaptive than it was previously. This occurs at both a domain level (the specific tier-one productions) and at a general level (the general tier-one productions). Klahr and Wallace see discontinuity arising though the gradual build up of knowledge until a certain point at which behaviour suddenly changes. That is, the culmination of a number of new productions enable behaviour to change suddenly.

2.5.1.2 Siegler

The theory proposed by Siegler and his colleagues focuses on strategies, and can therefore be said to be chiefly aimed at the cognitive development of problem solving. Detailed analysis of children's problem solving behaviour during the onset of cognitive change (the detail is obtained by the microgenetic method, Siegler, 1995) shows that any theory which describes how transitions occur during development has to account for five phenomena (Siegler & Shipley, 1995):

- Variability – Children's strategy selection shows that they use a variety of strategies for the same problem; even the same child uses different strategies for

the same problem over a short time-course (the subsequent strategies are not necessarily be more efficient).

- Adaptive strategy choices – Children's choice of strategy tends to be adaptive to the current problem: fast strategies that are prone to error are used mainly on simple problems; on difficult problems, children normally select a more time consuming strategy which is more likely to be accurate.
- Change – With age and experience, the relative frequency of strategy use changes, the effectiveness of strategy use changes (in terms of children's accuracy in executing a strategy), and new strategies are acquired. Strategy acquisition occurs even when a current strategy enables accurate performance on problems.
- Generalisation – Children can generalise strategies to new problems; generalisation can be speeded by presenting challenging problems.
- Individual differences – The distribution of frequencies of strategy use differs amongst children.

Following these findings, Siegler and his colleagues developed an Adaptive Strategy Choice Model (ASCM, Siegler & Shipley, 1995). Development in this model "reflects interactions among a developmentally invariant retrieval system, a changing representation of factual knowledge, a simple learning mechanism, and experience in the environment" (Kail & Bisanz, 1992, p.248). The learning mechanism records speed and accuracy information for every strategy (each time the strategy is used), aggregated over all problems, problems with a specific feature, and specific problems. The novelty strength of a strategy is also recorded, and is reduced every time the strategy is used.

When encountering a problem, the recorded data is used to project the likelihood of each strategy solving the problem, in terms of a projected strength for each strategy. The probability of selecting a particular strategy is proportional to its strength, relative to the strength of all the strategies combined. The execution of some strategies (those that are purely cognitive) results in their execution time being reduced, and their accuracy increased (if the strategy was successful).

The ASCM model can account for most of the phenomena listed by Siegler and Shipley. The probability factor is able to account for both variability and adaptive strategy choice. Several strategies each have an associated probability, meaning selection of strategies is variable. Strategy choice becomes adaptive because a successful strategy on specific problems will lead to its strength being increased over time, such that the model adapts to

be more likely to select the strategy for the problem in the future. By recording feature information, the model can generalise to apply strategies to problems having similar features to those problems on which the strategy has been successful in the past.

Although the model does not account for strategy acquisition, the relative use of each strategy changes with experience. Cognitive strategies are also executed quicker and more accurately with experience. Individual differences can be accounted for through a difference in the distribution of the strategy strengths across individuals. This represents the effects of different experiences with problems across children.

The ASCM model was able to fit the data of children in simple addition problems (the principles of the model have also been applied to other areas such as spelling, problem solving, and memory tasks, Siegler, 1991). Addition uses several different strategies, such as retrieval of the answer from stored knowledge of the problem, counting each addend on the fingers and then counting all fingers raised, and counting up from the largest addend. Although the latter strategy had to be added to the model after several runs (because the children acquired it during the experiment, and the model does not include strategy acquisition), the model still captured the adaptiveness, accuracy, and distribution of strategy use.

2.5.2 Context based theories

Vygotsky is the main proponent of the social contextualist approach to cognitive development. The approach focuses on the impact that social factors have upon cognitive development. A major contributor to development is therefore the help and support of both adults and peers (Flavell, Miller & Miller, 1993). The external world is not viewed as separate to the child – rather, the two have a strong relationship, where children behave in certain ways because the environment serves their current needs and goals (Miller, 1993). Any learning that occurs is therefore within the context of the external situation that instigated the learning.

Vygotsky believed that any cognitive function appeared on two planes, the first on a social plane through engagements between the child and the environment, and the second on a psychological plane, where the function is internalised (Vygotsky, 1981). There is therefore a close relationship between the internal structure of a child-environment engagement, and the actual engagement that took place. However, the internalised

functions are not a simple copy of the social process, and the internalised functions will be different depending upon the social process that took place (Wertsch & Kanner, 1992).

The use of internalised functions can be mediated by various processes, the main one being internalised speech. This allows functions to be more sophisticated, and makes incorporation of the function into human action easier and more efficient (Wertsch & Kanner, 1992).

The degree of cognitive change is limited to what Vygotsky (1978) described as the child's zone of proximal development. This represents the span from the present abilities of the child, to what he could achieve with help and appropriate instruction. This means that the zone is limited to the child's current intellectual capabilities, and the form of instruction that is involved. Proceeding through the zone requires children to build upon what they already understand. Development is therefore the gradual process of building upon what knowledge is already in place.

2.5.3 Capacity based theories

Three theories have been put forward by Pascual-Leone, Case, and Halford that each recognise capacity as a limiting factor of development. All of these theories are based upon Piagetian theory. Each rejects the concept of knowledge structures as being logic based, and each believes that Piaget neglected functional aspects of development, such as individual differences (Beilin, 1994). The theories provide knowledge based accounts, where the knowledge that can be accrued is mediated by the individual's capacity limitations.

2.5.3.1 Pascual-Leone

Pascual-Leone (1969) developed a theory which was heavily based on that of Piaget, attempting to retain its strengths and reject its weaknesses. Pascual-Leone kept the idea of knowledge structures, and their constant elaboration, together with assimilation and accommodation. However, several additions were made in order to combat the weaknesses of Piaget's theory.

An activation and weight is associated with each knowledge structure. The activation of a knowledge structure is mediated by its weighting. The weighting is based on a number of factors (Case, 1985): the number of cues for the knowledge structure that are present in the perceptual field; the effectiveness of the cues, in terms of their salience and organisation; the number of cues that activate related knowledge structures (if a

knowledge structure is a super-ordinate, the activation of its sub-ordinate knowledge structures means the super-ordinate structure can become active); the attentiveness of the subject.

When there are lots of cues that are associated with a knowledge structure, together with cues that are associated with related knowledge structures, and the cues are salient in relation to other cues, then it is very likely that an appropriate knowledge structure will be retrieved. Activation of knowledge structures is therefore linked to the number of cues that can be attended to in the perceptual field.

Pascual-Leone also introduced the idea of misleading task structure. Cues in the perceptual field, together with previous learning, may suggest knowledge structure *x*, when in fact knowledge structure *y* is appropriate. The theory proposed that an increase in mental attention is required to resolve the conflict amongst different knowledge structures. Another concept is therefore introduced: that of quantitative increases in mental attention (termed M-power). This represents the maximum number of knowledge structures that can be activated simultaneously (and can therefore assumed to be linked to working memory capacity). Stage transition therefore represents an increase in mental effort (or M-power) with age (Pascual-Leone, 1970), enabling tasks to be solved which, in the previous stage, required the simultaneous activation of too many knowledge structures.

The view held by Pascual-Leone is that it is misleading task situations which allow children to show age-related bursts of performance improvement (Pascual-Leone, 1987). The knowledge structures associated with the misleading information must be actively inhibited by the subject, and the task relevant knowledge structures must be boosted. An interrupt operator of central attention is proposed to achieve the inhibition (Pascual-Leone & Morra, 1991), which means the M-power can be diverted to increase the activation of related knowledge structures that are relevant to the task. An increased mental effort is required to achieve this, which is why stage transitions will often arise from misleading task situations (Pascual-Leone, 1995).

The M-power concept allowed the theory to cope with the problem of horizontal décalage, because the various tasks that were assumed to require the same logical concepts may require different amounts of knowledge structures, and therefore different levels of

M-power. This explained why problems having the same underlying logical structure may be mastered at different ages.

2.5.3.2 Case

The Case (1978; 1985) theory of development integrates both Piagetian and information processing theories of development, as well as that of Pascual-Leone. His theory is stage based and puts forward the idea of development occurring by the generation of successively more complex knowledge structures and strategies. The acquisition of these is mediated by both working memory capacity, and experience. Memory capacity strongly influences acquisition because the more complex knowledge structures and strategies require various dimensions of information to be integrated at one particular time. Experience can help in this endeavour because it enables some knowledge to be chunked, and some strategies to be automised, thereby freeing capacity resources.

The role of automatisation is of further importance because the theory denies the development of memory capacity per se. Rather, it is the functioning of the capacity which changes. Automatisation plays a crucial role by reducing the amount of capacity that the basic operations of a task may require, allowing attention to be focused on other aspects of the task (Case also allows the possibility of maturation to improve efficiency). Further experience of the task enables within-task aspects to become automatised and thus allows more efficient task strategies to be sought.

Memory capacity is conceptualised similarly to the M-power concept of Pascual-Leone. However, Pascual-Leone believed that the number of knowledge structures that could be activated simultaneously was chiefly influenced by the current M-power attainment, but could also be influenced by, for example, perceptual stimulation. Case views the number of knowledge structures that can be simultaneously active to be solely based on memory capacity. This means that intense perceptual stimulation will activate knowledge structures, but this will mean that the mental stimulation of knowledge structures must decrease, because the number of knowledge structures that can be activated remains constant (Case, 1985, p.289).

Transition across stages occurs by a process of hierarchical integration. When a sequence of knowledge structures are activated in a novel sequence, and a functional utility of this sequence is recognised, then the sequence is tagged so that it can be activated intentionally in the future. The practice of intentionally activating the sequence enables it to become a

new knowledge structure. The process is hierarchical because a new higher-order knowledge structure evolves from lower-order knowledge structures. Hierarchical integration enables a new stage of behaviour because the newly created knowledge structures are based on lower-order knowledge structures from a previous stage, whose form and function in the previous stage was considerably different.

Case proposes that each stage has three sub-stages (operational co-ordination, bifocal co-ordination, elaborated co-ordination). Movement from one sub-stage to another involves integrating knowledge structures. However, this does not have to be hierarchical.

Case's theory overcomes many of the limitations of Piaget. For example, horizontal décalage is explained by the acquisition of knowledge relating to a different *level* of a logical concept, rather than a different *type* of logical concept. This means that a concept can be acquired at any age (vast experience of one concept means its level is raised), whilst the Piagetian view requires the underlying knowledge structures related to the concept to be in place before the concept can be acquired. The view of horizontal décalage remains; only its interpretation changes (Case, 1985, p.255).

2.5.3.3 Halford

Halford (1993) puts forward three key areas to development: understanding, learning, and capacity. The key area is understanding, because it allows skills and strategies to be developed. In order to understand a concept, a mental model is required, which represents the structure of the current problem or situation. Representing the problem structure draws upon domain knowledge, which is both declarative (factual) and procedural (skills and strategies). Thus a part of understanding, which is necessary for the development of skills and strategies, draws upon skills and strategies itself. This suggests that new skills and strategies can incorporate those that have been learnt previously.

Mental models require domain knowledge (not necessarily from the same domain as the current problem). Knowledge must therefore be acquired so that understanding can be improved. Acquisition is attained through processes of learning. Halford separates learning into two parts: basic learning processes that acquire the knowledge that is needed for the building of mental models, and domain-general methods (such as analogy) that create domain-specific skills and strategies. The domain-general methods operate on the knowledge acquired by the basic learning processes, to generate skills and strategies.

The skills and strategies which use domain knowledge are limited by capacity (Halford, Maybery & Bain, 1986). Capacity is viewed as a finite number of dimensions rather than a finite number of items. Dimensions represent the number of independent units of knowledge that are required to represent a concept.

In Halford's definition of capacity, it is the amount of information that can be accessed *simultaneously* that is limited. As the number of dimensions increases, more information can be accessed in parallel, and therefore the complexity of the knowledge that can be represented increases.

Strategies can be learnt whereby limited capacity can be overcome. Halford suggests two strategies: Recoding domain knowledge such that it requires fewer dimensions, and segmenting domain knowledge into components, where each component requires fewer dimensions, and components are accessed serially. Capacity will only affect serial processing when the values of intermediate steps in the processing need to be stored (because this effectively adds another dimension).

Halford proposes that one-dimensional concepts can be represented when a child is one year old, two-dimensional concepts when the child is two, three-dimensional at three, and four-dimensional at eleven. The ability to recode concepts in fewer dimensions greatly improves efficiency because it allows other concepts to be represented by the remaining dimensions (allowing for dimensional limitations through capacity).

Capacity affects both understanding and learning, because it impacts upon the complexity of knowledge that can be represented. Not all learning mechanisms will be affected: the basic learning processes that enable knowledge acquisition are not expected to be mediated by capacity limitations, although the domain-general learning mechanisms, such as analogy, will be (Halford, 1995).

Halford's theory incorporates no real definition of stages: increased capacity enables the increased structural complexity of domain knowledge. This enables improved performance. However, because knowledge is acquired within domains, general improvements in performance (i.e. stage-like) are not expected to take place, although overall improvements should be seen at the ages when capacity increases. The theory suggests that more complex knowledge arises from that which is less complex, in line with

Piaget, but it is learning which governs structural complexity rather than accommodation and assimilation.

2.5.4 Processing based theories

The main proponent of processing speed as a mechanism of development is Kail (1996). Kail does not put forward a theory of development. He admits that processing speed is not the only developmental mechanism, but leaves the other mechanisms unspecified.

The reason for examining processing speed as a possible mechanism is due to the limited processing resources that information processing theory proposes as an influence upon development (Kail, 1991). Processing speed should interact with processing capacity, especially where processing capacity is limited, because processing speed will influence how quickly resources can be allocated to other tasks.

Several studies (e.g. Kail, 1986) indicate that reaction times decrease exponentially with age on a variety of tasks. This suggests a global mechanism that is influencing the speed of processing. Various ideas have been put forward as to why the speed of processing increases with age: the use of more efficient strategies; a larger knowledge base; an increase in processing resources; an increase in memory capacity. All of these are inconsistent with the findings of empirical studies:

1. Where there was no apparent difference in preferred strategy across ages, there was still a difference in task time (Kail, 1988). When comparing baseball novices and experts in the mental rotation of both familiar baseball players faces and unfamiliar faces, "greater knowledge did not consistently lead to faster processing" (Kail, 1991, p.162).
2. When performing two tasks at once, the increased load on processing capacity should mean the tasks take longer to complete than when attempted separately. Many studies do not show this prediction (Kail, 1991).
3. If changes in processing speed were due to changes in memory capacity, then in experiments involving age, speed, and memory, it is expected that correlations between age and speed will be significantly reduced if memory is partialled out. This is not the case (Kail, 1996). In fact, Fry and Hale (1996) found quite the opposite: that it is processing speed which influences memory capacity rather than vice versa. They also found that age has a strong relationship to processing speed.

Processing speed appears to be a plausible mechanism of cognitive development. It is considered to be a global mechanism because it influences performance on a variety of tasks. Speed may also change with age for motor actions (Pascual-Leone & Morra, 1991), and eye movements and fixations (Kennedy & Murray, 1986).

2.6 A summary of what develops

One problem of evaluating claims about what develops is that several theories propose very similar ideas and mechanisms of development, yet each uses different descriptions. For example, strategies are described separately to other knowledge in some theories (Siegler, Halford), in some theories a strategy is not described at all (Piaget, Pascual-Leone, Case), and in others, a strategy is called something completely different, such as a production rule (Klahr and Wallace). The reason for this is twofold: first, the level of detail across theories differs; second, explanations of terminology are often omitted. This is why, where definitions are difficult to compare, the general term knowledge structure is used.

An account of the ways in development occurs in each theory is summarised in Table 2.1. Differences in the level of detail of the theories can be seen easily; for example, all Siegler's mechanisms can be seen as changes in knowledge, as can those of Klahr and Wallace. Note that the connectionist and information-processing theories are omitted here as they are frameworks from which to base theories. For example, connectionist networks can have various learning mechanisms; a network using learning mechanism "A" represents a slightly different theory to a network using learning mechanism "B".

Table 2.1: The methods of development that are proposed by theories of development.

Theorist	What develops	Process of development
Piaget	Knowledge structures	Accommodation; Assimilation
Klahr and Wallace	Novel productions; Productions based on existing productions which are either more specific or more general	Storage of timeline; Examination of timeline for consistent sequences and redundancy elimination
Siegler	Strategy choice; Strategy generalisation; Strategy acquisition; Strategy efficiency	Strategy speed; Strategy accuracy; Strategy novelty
Vygotsky	Knowledge structures	Engagements with the environment; Instruction
Pascual-Leone	Knowledge structures; M-power	Maturation; Accommodation; Assimilation; Inhibition of misleading knowledge structures
Case	Knowledge structures; Functional memory capacity; Strategy acquisition	Experience; Maturation; Automatisation
Halford	Domain knowledge; Skills and strategies; Dimensional capacity	Knowledge acquisition; Learning; Maturation;
Kail	Speed of cognitive processing Possibly also speed of: Motor actions Eye movements and fixations	Neural mechanisms

2.7 What is wrong with the theories?

The above list of theories of development is not exhaustive. There are two main reasons why there are so many theories of development: the theories are vague and the theories are difficult to test. Each is considered in turn.

2.7.1 Lack of precision

The lack of precision is not solely the fault of the theorist. The developmental data itself is not ideal. Development occurs over many years, so obtaining a good quantity of developmental data even on a single task is difficult. Given that most theories are based on empirical data, if the data itself lacks clarity then it is no wonder that the theories themselves also lack clarity.

Development has tended to be studied cross-sectional rather than longitudinal. The studies therefore present children's task performance at macro-levels (across ages), rather than their task performance at micro-levels (within and across ages). The resultant data means that it is difficult to create precise theories of development because the theories have to bridge a gap in the data that the empirical studies have left. This problem has been well documented by Siegler (Siegler, 1995; Siegler & Shipley, 1995).

The microgenetic approach to studying development (Siegler & Jenkins, 1989) has been proposed to combat the lack of longitudinal data. The approach helps the examination of mechanisms of development by concentrating task studies on a period when a change in task performance takes place. The approach helps to detail possible mechanisms of development that occur within a limited time span (the period when the change in task performance is observed). The narrow time span means that the applicability of the microgenetic approach to the study of development in general is difficult to assess. Nevertheless, Siegler and colleagues have used the approach across domains and have found similar mechanisms can explain changes in task performance (Lemaire & Siegler, 1995; Siegler & Shrager, 1984).

Another contributor to the lack of precision in the theories is that most of them are verbal theories. Verbal theories are expected to be vague in parts, which is not helped by the problems of the developmental data that are outlined above. For example, it is widely acknowledged that Piaget's mechanisms of accommodation and assimilation lack precision (e.g. Flavell et al., 1993; Miller, 1993) and are therefore difficult to apply to specific domains. The lack of precision that is indicative of verbal theories leads to different interpretations of the theories. The interpretations may not necessarily be representative of the theorists intentions.

2.7.2 Difficult to test

There are two reasons why the theories are difficult to test. First, their imprecise nature leads to the theory having to be interpreted. As stated above, the interpretation may not be in line with the theorists intentions. Second, the mechanisms specified by the theories are difficult to test empirically because it requires all other mechanisms being controlled for. How can knowledge be controlled for in a theory which includes capacity when the two are likely to heavily interact? How can you be certain that a child is at capacity *x* and not capacity *y*? How can you be certain that children in studies have understood the task presented to them, and therefore the conclusions to the study are valid? Testing mechanisms empirically raises so many critical issues that it is difficult to rely on the findings of studies without support from other sources (such as computational modelling). Researchers such as Halford and Siegler now present implementations of their mechanisms within computational models as support for their theory.

2.8 How can computational modelling help examine development?

The two central problems of theories of development can both be overcome to a large extent by implementing the theories within a computational model. A computational model forces precision because it is a computer program which usually requires a full specification of how processing is to be accomplished (i.e. how the developmental theory is to be implemented). This forces a decision regarding any imprecise parts of the theory, leading to a more descriptive theory.

Computational models can provide a rigorous test of the mechanisms that are specified by developmental theories. When several mechanisms are implemented within a single model, one mechanism can be manipulated whilst keeping all others constant. In this way the effect on behaviour of single mechanisms of development can be identified.

2.9 Summary

This chapter has detailed several theories of development and has explained the problems of the theories. Computational modelling has been put forward as a method for overcoming the problems, and in particular as a method for testing theories of development. The next chapter describes various models of development so that it can be seen how many of the mechanisms of development that are described in Table 2.1 have been implemented within a computational model. This will reveal the extent to which theories of development have been tested within a computational model.

3. Computational models of development

The previous chapter detailed theories of development, and the mechanisms of development that are included in the theories. This chapter details computational models of development. The aim of the chapter is to see to what extent the theoretical mechanisms of development described in Chapter 2 have been implemented within computational models of development.

The chapter begins by examining models of the balance-scale task. The balance-scale task is used by several models, and these models span almost all types of modelling architecture. It is therefore an ideal task to describe in detail, together with the models that have been developed to match subject data on the task. Other models of development are then detailed so that the full spectrum of developmental mechanisms within models is covered. This will aid a comparison between mechanisms of development proposed by developmental theory, and mechanisms of development that are implemented in computational models. A general critique of the models is then given, which shows that the mechanisms of development used by the models do not cover the array of developmental mechanisms that theories of development propose. The chapter ends by proposing the testing of developmental mechanisms within a computational framework.

3.1 Models of the balance-scale task

There are two reasons for describing several models that model the same task. First, the task need only be described once. Second, modelling the same task means the models can easily be compared, especially the mechanisms by which development occurs within them. The task itself will be described, followed by the task phenomena, the models of the task, and a comparison of the models.

3.1.1 The balance-scale task

The balance-scale task was first described as a developmental task by Inhelder and Piaget (1958). The task has been well documented and so only a brief description is given here. A seesaw balance is used, which has a crossbar with a fulcrum at its centre. Weights can be placed on the crossbar, on either side of the fulcrum and at different distances from the fulcrum. This enables the crossbar to either remain horizontal, tip left, or tip right. The key concepts are the size of the weights that are placed on the fulcrum, and the distance they are placed from the fulcrum. Correctly co-ordinating these two concepts (by means

of computing the torque on the crossbar) means that the movement of the crossbar can be calculated accurately.

3.1.2 Data from the task

The most rigorous analysis of data on the balance-scale task data is provided by Siegler (1976; 1981). Siegler found that children's behaviour could be characterised by a set of four progressive rules which correspond to stages in performance on the task:

1. Use only weight information to judge balancing.
2. Same as rule 1, but take into account distance when there are equal weights on either side of the fulcrum.
3. When both the weights and distances are equal, then the crossbar will balance. Otherwise, if the weights are different but the distances are equal, then only take the weights into account. If the distances are different but the weights are the same, only take into account distance. When both weights and distances are different, then guess.
4. Compute the torque on the crossbar based on the weight and distance of items on both sides of the fulcrum. Computing the torque is always accurate.

Siegler systematically analysed task behaviour by creating six problem types, each of which yielded a specific probability of success for the different rules. Balance problems have the same weights at the same distance. Weight problems have different weights but at the same distance from the fulcrum. Distance problems have the same weights at different distances from the fulcrum. The other three problem types are where both weights and distances are unequal. In conflict-balance problems, the crossbar remains horizontal. In conflict-weight problems, the side with the most weight goes down. In conflict-distance problems, the side with the most distance goes down. The six problem types enable absolute predictions for the accuracy of each of Siegler's rules. These are shown in Table 3.1.

Table 3.1: Percentage accuracy for each of Siegler's rules in the balance-scale task. Chance-level performance is at 33% accuracy.

Problem type	Rule 1	Rule 2	Rule 3	Rule 4
Balance	100%	100%	100%	100%
Weight	100%	100%	100%	100%
Distance	0%	100%	100%	100%
Conflict-balance	0%	0%	33%	100%
Conflict-weight	100%	100%	33%	100%
Conflict-distance	0%	0%	33%	100%

Using these problem types, Siegler found that the large majority of children aged between five and seventeen behaved in a way which conformed to the behaviour that is characterised by the use of one of the four rules. The behaviour of the majority of five and six year old's could be characterised by rule one. The behaviour of children aged between nine and ten years could be characterised by either of rules two and three. The behaviour of children of thirteen years and above was mainly characterised by rule three, although some of the older children's (16-17 years) behaviour could be characterised by rule four.

The more complex rules were used by older children, and the children seem to progress, with age, in the order of the rules. The youngest and oldest children are interesting: the youngest children all know about distance yet fail to take it into account when attempting to solve the problems; many of the oldest children do not reach the most complex level of performance. Asides from the general results, these are aspects of the task behaviour that will also need to be addressed by any model.

3.1.3 Models of the task

The first task model was devised by Klahr and Siegler (1978), and was a production system which contained production rules that mapped on to the rules elicited by Siegler for the task. The behaviour at each of Siegler's rules was produced by adding more complex production rules to the production system (this method has been used in other domains such as seriation [Young, 1976]). The method by which behaviour was modelled can therefore be seen as the introduction of new knowledge into the system in the form of production rules: beginning with the production system for Siegler's first rule, each subsequent rule was modelled by taking the previous production system and adding production rules to it. The production systems were able to account for a lot of the

variance in children's behaviour on the task, but they do not explain how the rules are developed.

Langley (1987) devised a production system model of the task which began with production rules that gave random behaviour, but through learning, the production system accounted for the behaviour seen for the first three rules stated by Siegler. The learning process strengthens rules that are successful, and when a rule is unsuccessful, looks for discriminations between itself and successful rules. The discrimination process results in the creation of a new production rule.

The Langley production system is able to generate the rules that provide the transitions in behaviour in the Siegler study. However, it also generates rules that are not applicable, for behaviour that is not seen on the task. For example, distance rules may be generated as part of the first stage, when children only take into account weight. This is why the model's behaviour is not compared to subject's behaviour. Langley recognises the problem of the generation of incorrect rules but does not suggest how to overcome it (one applicable method used by later models of the task is to bias the problem types presented to the model). The model therefore provides a transition mechanism for the task, but fails to provide a match to the subject data.

Newell (1990) details a balance-scale model written in Soar, a cognitive architecture realised as a production system language which is able to learn new production rules through a chunking mechanism. The model has three processes: one to encode the problem, one to compare values of the encoded dimensions, and one to select whether one side of the crossbar will tip, or whether the crossbar will balance. Learning takes place when the model encounters an impasse (when it does not know what to do next, such as having three choices all of which are equal in value), and whenever the model's solution to a balance problem is incorrect.

Initially, Newell's model starts off with very little knowledge, and will select a solution to a balance problem by selecting a random alternative from the three possible outcomes: left down, right down, and balance. The lack of knowledge for selecting an outcome represents an impasse. A new production rule is learnt from the impasse. The new rule means that the randomly selected outcome will be used on every subsequent balance problem, unless the model receives negative feedback, or new rules are created.

If the model receives feedback indicating its selection was incorrect, then the model will attempt to discriminate. This means either attempting a new form of comparison with the task dimensions that are known to the model, or seeking a new task dimension to encode. This leads to a more specific production rule than was previously known.

Given an ideal sequence of trials, the model is able to learn up to Siegler's rule three after four trials. The model never learnt rule four, though it is not clear whether this is because it could not, or whether it was not of interest to Newell. The fact that the learning of rule four is omitted would suggest that the model failed to learn it.

McClelland and Jenkins (1991) created a connectionist model of the balance-scale task, which consisted of twenty six units. They assumed a balance-scale which had five pegs on either side of its fulcrum, and weights ranging from one to five. The network had twenty input units, five for the left distance and five for the right distance, five for the left weight and five for the right weight. The distance input units were fed into two hidden units, and the weight input units were fed into two separate hidden units. The hidden units of the architecture therefore represented distance and weight separately. All four hidden units fed into two output units, one representing the left side of the crossbar dipping and the other representing the right side of the crossbar dipping. A balance occurred when the activation difference between the output units was less than 0.333.

The network started with all weights randomised to between -0.5 and +0.5, and was trained with balance problems which were predominantly weight based (the weight dimension on both sides of the fulcrum differed more often than the distance dimension). The network was trained for 100 epochs.

The initial response of the network was for the crossbar balancing, although after approximately 15 epochs the network's behaviour corresponded to Siegler's rule one. The performance of the model became gradually more complex such that its behaviour mapped onto each of the rules described by Siegler, and in the correct order (although rule four was never mastered by the network, it's performance often mapped on to rule four). The behaviour of the model during training was variable such that in most cases, for example, behaviour could be characterised by rule two, but sometimes could be characterised by rule three. McClelland and Jenkins state this to be consistent with the test-retest data reported by Siegler (1981).

Shultz, Mareschal, and Schmidt (1994) detail three connectionist networks of the balance-scale task which build up their own hidden units by the use of a cascade-correlation learning algorithm. Cascade correlation (Fahlman & Lebiere, 1990) begins with a minimal connectionist architecture (input and output units only) and adds hidden units to the architecture at specified intervals in the network's training (when the output error no longer decreases sufficiently). The hidden unit that is added is selected from a pool of units that all undergo training with the input data; the unit that has the best correlation with the output error is selected. The network is therefore able to generate additions to its own internal architecture when necessary.

The first of the three networks will be covered here, because the two other networks only vary slightly from it. All networks had four input units and two output units. The four input units coded weight and distance for both sides of the fulcrum, where weight and distance varied from 1 to 5 for each unit. The output units were used the same way as McClelland and Jenkins. The training set was significantly biased towards weight problems. The training set was expanded whilst the output error was decreasing sufficiently, by one training example on each cycle. Each training example added to the training set was subject to the same weight bias, and reflected an assumption that the child's learning environment gradually changes. Sixteen sample networks were trained for 300 epochs each.

The results of the network show a gradual improvement in performance that can be characterised as the network progressing from Siegler's rule one through to rule four, in the correct order. The network is usually able to master rule four, although training has to be extended beyond 300 epochs for this to occur in some of the networks. Between two and three hidden units, on average, are added to the network. The majority of hidden units were sensitive to the information of one side and one dimension: for example, being sensitive to right side weight inputs, but the opposite for left side weight inputs. There is some variability in the behaviour of the network, as there was with the McClelland and Jenkins model.

3.1.4 Comparison of the models

The exercise in detailing several models of the same task is not to show the advantages of one learning methodology over another (e.g. connectionist, or sub-symbolic, versus symbolic modelling). It is apparent that the connectionist models perform better on the balance-scale task but these models are the most recent, so this is expected (there is no

reason to publish a recent model which is no better than previous ones). There is a recent symbolic model that achieves similar success to the connectionist models (Gobet, in press), but this was not described because of insufficient detail. The models are now compared on a variety of factors. The comparison is shown in Table 3.2.

The initial behaviour of each model is important because this corresponds to the model's prediction of children's behaviour before they show rule one performance (if the model does not start at rule one). In the Siegler study, 77% of five and six year old's behaviour could be characterised by rule one, yet 23% of children could not be classified into a rule. The behaviour of these children may be interesting to examine, as well as the behaviour of children below five years, because this is the pre-rule-one behaviour. As Table 3.2 shows, the predictions for this behaviour differ across models.

Table 3.2: Comparison of computational models of the balance beam. Entries with a * indicate the author gives no explanation for the phenomena, but one is given here based on the way the model learns.

	Klahr and Siegler	**Langley**	**Newell**	**McClelland and Jenkins**	**Shultz et al**
Starting behaviour	Rule one.	Random guesses.	Random guesses.	Predicts balance for ~fifteen epochs.	Not stated, but likely to predict balance.
Ending behaviour	Rule four.	Rule three.	Rule three.	Rule four.	Rule four.
Ability to match subject behaviour	Good. Matches stage behaviour by using different production rules.	Average. Matches stage behaviour by using different production rules, but omits Siegler's rule four.	Average. Matches stage behaviour by using different production rules, but omits Siegler's rule four.	Good. Matches stage behaviour by having more training on balance problems, shows variability that subjects show (flipping between rules two and three), but never really masters rule four (some subjects do).	Very good. Matches stage behaviour by having more training on balance problems, shows variability that subjects show (flipping between rules), and masters rule four.
Method of transition	N/A	New production rules, strengthening of successful rules.	New production rules.	Continual experience with the same balance problems.	Experience with balance problems, an expanding training set, and recruitment of hidden units.

	Klahr and Siegler	Langley	Newell	McClelland and Jenkins	Shultz et al
Speed of transition	N/A.	Quick.	Immediate.	Slow (100 cycles in all).	Very slow (300 cycles in all).
Explanation of weight bias	None.	Bias for weight in the training.*	Bias for weight in the training.*	Bias for weight in the training.	Strong bias for weight in the training.
Explanation for relatively few subjects acquiring rule four	None.	Model fails to learn rule four - explanation therefore must be a failure of the learning mechanism.	None.	Model fails to regularly use rule four - the explanation is therefore that learning to this complexity is difficult.	Insufficient experience with balance-scale problems.

The McClelland and Jenkins, and the Shultz, Mareschal, and Schmidt models progress through the four stages over the course of 100 or more training trials. This period bears much more similarity with children's behaviour on the balance-scale than the rule learning models, which are able to progress very quickly through the stages. However, there are rule learning architectures, such as ACT-R (Anderson, 1993), which are able to produce a more gradual shift to the new rule by implementing rule strengths. This would also introduce the variability in behaviour shown by some of the other models, and by some subjects. Newell acknowledges that the rule strength approach could have been used in his model, but used discrimination learning because it was the most natural for Soar.

Siegler found that the behaviour of children of ten years and below could often be characterised as being biased by weight, and ignoring distance. All of the models explain the initial weight bias by having more weight based problems in their training set, often stating that this simulates children's experience with weight being far more extensive than their experience with distance. This appears to be an unsatisfactory explanation. Children at the age of five know about distance (Newell, 1990), so the obvious explanation for the weight bias is that children *encode* only the weight dimension. Only the Newell model encodes dimensions, although the model does not explore the process. The other models are tapping into the fact that the children attend to weight before distance (Plunkett &

Sinha, 1992, give this explanation for the McClelland & Jenkins model). They do not explain the process in terms of encoding. This may be a flaw in these models, because children focus on distance rather than weight in other types of balance task (Karmiloff-Smith & Inhelder, 1974).

There are further comparisons and general issues that could be reported from these models. Each will also be applicable to the models that will be discussed in the next section. Full discussion is therefore left until the final section of this chapter.

3.2 Models of other developmental tasks

Computational models of other developmental tasks are now described. The focus will be to present models which include developmental mechanisms, but which model tasks other than the balance beam. The tasks modelled are the conservation of number, the acquisition of the sum-to-min strategy, transitive inference, relational similarity, and the concepts of velocity, time, and distance. Models of other task domains could be described (e.g. language acquisition), but the models that will be described here are sufficient to present all of the developmental mechanisms that have been incorporated into computational models of development.

3.2.1 Conservation of number

T. Simon and colleagues (Simon & Klahr, 1995; Simon, Newell & Klahr, 1991) detail a Soar model of the conservation of number, which is based on a study by Gelman (1982). Conservation involved two rows of counters arranged in a one-to-one mapping; one row is transformed (e.g. by increasing the space between the counters) such that the one-to-one mapping is no longer preserved; the children are asked if the number of counters in the rows are the same.

The Gelman study examined the effect of training on performance; a training phase was given, followed by a standard conservation test. In training, groups of three and four year old children were given three different levels of help: none, some (children were asked to count the items in one of the rows), or extensive (children were asked to count the items in both rows, and compare the results). In the post-test, only the three year olds given extensive help performed well; the four year olds receiving any type of help performed well.

Based on these findings, Simon and Klahr (1995) detail five assumptions which they believe govern the conservation performance in the Gelman study.

1. If a row is not transformed, the quantity of items in the row remain the same.
2. If a row is transformed its item quantity is likely to change.
3. If measurement is possible, four year olds can verify whether (2) is true or false.
4. Three year olds are only motivated to test the effect of a transformation if they are faced with conflicting sources of evidence.
5. Both three and four year olds can store and recall pre and post-transformation results, but only four year olds do so systematically.

Using these assumptions, Simon et al developed a production system model of three year olds performance on conservation, and a model of four year olds performance on conservation. The difference between the models is the production rule knowledge they have, derived from the assumptions given. The models show the same performance in conservation as the respective ages of children. Although the performance difference between the three and four year olds is indicated by the basic assumptions (particularly assumptions 3-5), by providing an implementation of the assumptions, they have been shown to be able to model the children's data.

The Gelman three year olds learn from being given extensive help in their training; the four year olds learn from being given either level of help in their training. The models also learn to do conservation. Both models begin as non-conservers. As Soar encounters an impasse, it learns new production rules based around conservation. The new rules are immediately available for use. A series of new production rules are created during the training of the models, such that on the post-test, the models match the behaviour of the children. The main difference between the two models is that the one modelling three year old behaviour does not have knowledge that the effect of a transformation can be verified by comparing pre and post-transformation values.

The Simon et al models imply that knowledge differences are the reason for the performance differences in conservation between three and four year old children. The model of the behaviour of three year old children has different knowledge (i.e. different production rules) than the model of the behaviour of four year old children. However, the reason that three year old's do not often store and recall pre and post transformations may be linked to capacity (as capacity is cited by many as a developmental mechanism, see Chapter 2). In the model the failure of three year olds to store and recall pre and post transformations is implemented as a failure to carry out a memorisation process (because

the model does not know that the effects of a transformation can be verified). The model ignores the possibility that three year olds do not store transformations because of capacity limitations. The models explain development only in terms of knowledge differences.

3.2.2 Acquisition of the sum-to-min strategy

The ASCM model developed by Siegler and colleagues (see Chapter 2) was developed based on studies of the sum-to-min strategy acquisition in children's simple addition. The model has also been successfully applied to other domains. For example, its predictions in the domain of multiplication have been confirmed. Lemaire and Siegler (1995) found that children used a variety of multiplication strategies, strategy use was adaptive based on problem types, and the distribution of strategy use changed over time. However, whilst the model can account for all of the changes in the use of strategies, it is not able to account for the acquisition of strategies.

Siegler and Jenkins (1989) found that four strategies exist in the transition from the sum strategy (counting both addends) to the min strategy (counting upwards from the largest addend). Jones and VanLehn (1991) developed a General Inductive Problem Solver (GIPS) model which starts with the sum strategy and is able to acquire the other three strategies, resulting in the min strategy. Siegler and Shipley (1995) state that this model may be able to overcome the limitation of the current ASCM model.

GIPS is based on the rating of operators via sufficiency and necessity values, enabling the model to select an appropriate operator. If the operator ends up being successful, then the values are altered to increase the chance of selection on a subsequent occasion; if it failed, the values are altered to inhibit selection.

Transition occurs through the changing of the pre-conditions for an operator. The pre-conditions of an operator are only modified once the model has sufficient experience in using the operator. When this is the case, pre-conditions for the operator can be added or removed. If a specific literal is always present when the operator is successful, then it is added to the pre-conditions of the operator. If an operator is applied when only some of its pre-conditions were matched, the unmatched pre-conditions are removed from the operator.

Whilst the model does not attempt to fit the data from Siegler and Jenkins (1989), it is consistent with some of the Siegler and Jenkins findings. For example, children were not observed to attempt strategies that did not conform to the principles of addition; GIPS does not produce any illegal strategies either. Jones and VanLehn state that incorrect strategies would be created were it not for the feedback that the model receives on whether an addition was correct or not. This demonstrates that good feedback is sufficient to create the observed behaviour on the task. The model predicts that children show no signs of developing illegal strategies because of the feedback they receive.

The model enables transition using two processes: automatisation and generalisation. Performing the task several times (i.e. automatising the task) leads the model to try and generalise its behaviour, by either adding or removing conditions to make subsequent processing more efficient. Removing conditions makes behaviour more general, and adding conditions enables behaviour to change over time.

There are four basic problems with the model. First, its performance has not been compared with subject behaviour. Second, an intermediate strategy is included which helps the sum-to-min transition take place. Children do not use the included strategy. Third, children sometimes skip strategies, which the model cannot do. Fourth, not all addition strategies are modelled. Siegler and Jenkins detail eight addition strategies that children use; Jones and VanLehn only cover the four that are involved in sum-to-min strategy acquisition.

3.2.3 Transitive inference

Halford et al. (1995) detail a model of transitive inference which acquires strategies based on the knowledge that the model has. Transitive inference involves deriving an ordered set from a list of premises, such as a>b, b>c, a>c => a>b>c. In the example, the third premise is redundant because the ordered set can be derived from the first two premises. In some cases, the ordered set cannot be derived from the first two of three premises. For example, in the premises a>b, a>c, the relationship between b and c cannot be determined until a third premise, b>c, is given. These are indeterminate problems which usually require relations across three elements to be considered simultaneously, rather than relations across two elements. The indeterminate problems therefore usually impose more processing load than standard problems if they are to be solved accurately.

Children have difficulty performing transitive inference before the age of five (Halford, 1989). In Halford's theory (see Chapter 2) this is because three dimensions (which are required by indeterminate problems) can only be processed by children of five years and over. The model provides an instantiation of the theory.

The transitive inference model is developed in the PRISM II production system language (Ohlsson & Langley, 1986). The model begins with domain-general production rules, and learns domain-specific production rules (which correspond to strategies) during run-time. Analogical reasoning is employed to determine the correct ordering of premises; general list manipulation operators are then used to place the premise elements in the correct order. For example, after the premises a>b, b>c, analogical reasoning maps the premises to an ordered set abc that resides in long-term memory. The operator Append is applied to arrange the premise elements ab and bc into the correct order. A domain-specific production is learned based on this result.

Domain-specific productions are used ahead of domain-general productions whenever possible. When a domain-specific production is successful, its strength is increased; when it fails, its strength is weakened. The strength of domain-general productions remains constant. All production strengths are subject to an amount of noise. The strength of productions needs to be above a specific threshold in order to be used. When several productions can be used, all of which are above threshold, the one with the highest strength is executed.

The mapping provided by analogical reasoning is important because it is influenced by capacity: A concept of effort is used in the model, whereby minimum effort means that mappings involving two element relations will be done, and maximum effort means that mappings involving three element relations will be done. The model begins with minimum effort, but effort is increased every time a production rule results in failure.

The model was run using twelve three-premise problems. The model quickly learns domain-specific production rules. The domain-specific productions do not always lead to success because initially they are created based on mappings of two element relations. The failure of some of the productions leads to the effort level being gradually increased until it is high enough to manage mappings involving three element relations. This enables further domain-specific productions to be created. The performance of the model on the twelve problems was consistent with children of nine years and over (for overall

performance; measures such as solution times and learning rates were not examined). The assumption that analogical reasoning is used in developing ordered sets was also supported.

The model is able to learn strategies and adapt its learning so that the most successful strategies become the most likely to be used (based on the manipulation of production rule strength by success and failure). Different levels of capacity are examined indirectly, by the gradual increase of effort when production rule strategies fail. However, it is clear that the model's performance involves more errors when only two element relations can be considered (low mental effort, equivalent to low capacity) than when three element relations can be considered (high mental effort, equivalent to high capacity).

Two issues are left outstanding. First, it is unclear whether five year old children could solve the inference problems that were presented to the model, and whether they showed similar errors to those of the model. Second, the model should show virtually error-free performance after it has encountered several problems, because appropriate production rules will have been created. It is not clear whether children also show error-free performance after practice on only a limited amount of problems.

3.2.4 Relational similarity

Gentner, Rattermann, Markman and Kotovsky (1995) detail a model of relational similarity. There is a general agreement that relational similarity is initially based on object-level similarities (e.g. the objects are both circles of the same size), and later shifts to relational similarities (e.g. even though the objects are different sizes, they are both the intermediate sized circles in their respective groups of three circles). The nature of the shift is believed to either be linked to cognitive stage (e.g. Halford, Maybery & Bain, 1986; Inhelder & Piaget, 1958) or to knowledge differences (e.g. Brown & DeLoache, 1978). Gentner and colleagues detail both a set of studies and a Structure-Mapping Engine (SME) model to show that the shift can be explained by knowledge differences.

Gentner and colleagues believe that relational similarity is largely based on finding the most structurally consistent mapping between representations. This belief is implemented in, and is best described in terms of, the SME model. Knowledge in the model consists of entities (e.g. circle), attributes which are predicates and describe an entity (e.g. the circle is red), functions to describe certain attribute types which cover dimensional properties (e.g. the circle has a size), and relations which are predicates

representing links between two or more entities, attributes, functions, or relations (e.g. the size of circle A is greater than the size of circle B).

Consider a problem involving two sets of three circles, where each set of three circles decrease in size from left to right. The size of the centre circle in the first set of three is the same size as the left circle in the second set of three; no other circles match for size. The goal of the problem is to decide which circle in the second set of three maps onto the centre circle of the first set. There are two realistic choices: the left circle, which matches on size (i.e. selection is based on object-level similarity), or the centre circle, which matches on relational structure, because it is the intermediate size of a group of three different sized circles (i.e. selection is based on relational similarity).

Adult subjects on tasks of the above problem type use object-level similarity to map entities. However, if subjects are given a pre-test which involves rating the pairs of sets for relational similarity, they are significantly more likely to use relational similarity to map entities. Gentner and colleagues found that the SME model shifts from object-level similarity mapping to relational similarity mapping when the representations of entities is made richer (by increasing the number of attributes for entities). Richer representation enables the SME model to find more relational mappings between entities. The shift shown by the SME model supports the experimental findings, assuming that the pre-test increases the knowledge (or the awareness of the knowledge) regarding the entities involved. Nevertheless, the SME model supports the general finding that an increase in knowledge leads to an increase in the probability that mappings will be based on relational similarity.

A similar type of problem was used to examine children's performance, although the objects used were changed to be more suitable for children, and were more varied in their predicted difficulty. Older children (four and five year olds) were more likely to use relational similarity for mappings than younger children (three year olds) in all conditions. However, when the three year olds were taught to give individual labels to the three entities in a set, their performance improved to be comparable to five year olds.

The SME model was able to simulate the first set of results by assuming that the three year olds failed to code relations between sizes of objects, whereas the five year olds were able to code size relationships. The second set of results is simulated by assuming that the teaching enables three year olds to code size relations (through the use of labelling).

The SME model again shows that an increase in knowledge leads to an increase in the probability that mappings will be based on relational similarity.

3.2.5 The relation between the concepts of velocity, time, and distance

Buckingham and Shultz (1994) detail a connectionist model based on the cascade-correlation learning algorithm which examines three related concepts: velocity, time, and distance. When children are given two of the concepts and asked to infer the third, Wilkening (1981) discovered that the children's behaviour was different depending upon which concept was being inferred. When inferring distance, both the five and ten year old children used the multiplication rule (d=vt). When inferring time, five year olds used a subtraction rule (t=d-v), and ten year olds used a division rule (t=d/v). When inferring velocity, five year olds used an identity rule (v=d) whilst ten year olds used a subtraction rule (v=d-t). Multiplication and division rules are the correct rules to use in the inference tasks.

The poor performance on velocity inference tasks was explained by memory demands: time information had to be held in memory. Wilkening (1982) reduced memory load for time by visually presenting time information. However, no differences were seen for velocity inference tasks across the two studies.

Buckingham and Shultz found that when the network had no hidden units, the behaviour of the network on all types of inference could be characterised by the identity rule. When one hidden unit was added to the network, its behaviour could be characterised by addition/subtraction rules. An additional hidden unit allowed the network's behaviour to be characterised by the multiplication/division rule for velocity and time inferences, with distance inferences requiring an additional hidden unit.

The network is fairly uniform in its performance. The addition of a hidden unit, in general, allows the model's behaviour to improve in a stage-like manner across all types of inference. The five and ten year old children do not show this type of behaviour.

To simulate the increase in memory demands for time information in velocity inference tasks (that Wilkening hypothesised), noise was added to the incoming time information. The network's behaviour changed such that at the same stage, different types of inference could be characterised by different types of rule. Memory demands changed the

behaviour of the network whereas Wilkening found no difference in children's performance.

Both types of network (the standard and the limited memory models) did not capture the behaviour of children acquiring the three concepts. However, the models showed that the three distinct rules governing acquisition (identity, addition/subtraction, multiplication/division) could be captured within a simple network architecture. In addition, the network showed that the acquisition of some concepts could be delayed by subjecting incoming information to noise.

3.3 Assessment of the models

There are four important aspects to discuss regarding the models of development detailed in this chapter. First, the number of mechanisms proposed by developmental theories that have been included in the models highlights areas of theory which have been ignored. Second, the large number of modelling researchers who have no background in developmental psychology provides a reason for why many theoretical mechanisms of development are ignored within computational models. Third, the models can be criticised because they only cover a small range of measures when matching subject data. Fourth, the problems of implementing developmental mechanisms within computational models is highlighted by two models which implement the same developmental mechanisms in different ways.

3.3.1 The imbalance between model mechanisms and theoretical mechanisms

The models presented above include a variety of mechanisms by which development occurs. The developmental mechanisms that are proposed by theories of development were listed in Chapter 2. Table 3.3 repeats that list, but maps each model according to the mechanisms that each model includes.

Table 3.3: The types of mechanism used by computational models of development. *The mechanism is also accounted for by knowledge structures.

Mechanism specified by developmental theory	Model(s) using the mechanism
Knowledge structures	All models
Novel productions*	Klahr and Siegler; Langley; Newell; Halford et al.
Productions based on existing productions which are either more specific or more general*	Langley; Jones and VanLehn
Strategy choice	Langley; McClelland and Jenkins; Shultz et al.; Halford et al.; Buckingham and Shultz
Strategy generalisation	Jones and VanLehn
Strategy acquisition*	Langley; Newell; McClelland and Jenkins; Shultz et al.; Jones and VanLehn; Halford et al.; Buckingham and Shultz
Strategy efficiency	None
M-power	None
Functional memory capacity	None
Dimensional capacity	None
Speed of cognitive processing	None
Speed of motor actions	None
Speed of eye movements and fixations	None

As Chapter 2 noted, there is overlap between some of the proposed developmental mechanisms (denoted with a *). All models develop their knowledge structures. The only model which is not explicit in increasing its knowledge base (i.e. new objects are not added to the model) is that of McClelland and Jenkins. The McClelland and Jenkins model adjusts weights between units, but does not add any new units to its architecture. However, both the weights between units and the units themselves are part of the model's knowledge. Adjusting the weights is therefore changing the knowledge base of the model.

The Langley model and the Halford et al. model enable adaptive strategy choice by manipulating the strength of production rules based on success and failure. The strategies (or production rules) which result in success become increasingly more likely to be

selected for use as the model performs the task. The McClelland and Jenkins, Shultz et al., and Buckingham and Shultz models enable adaptive strategy choice by giving the model more training on task specific examples. Some of the other models include changes in strategy choice, but this is accomplished by manually inserting new knowledge (e.g. a new production rule corresponding to the new strategy). This does not meet the definition of strategy choice in a model because the strategy choice mechanism is outside of the model.

The Jones and VanLehn model enables strategy generalisation because the model alters pre-conditions such that the conditions to satisfy for applying operators become more general or more specific. However, whilst this mechanism exists in the model, the behaviour of the model was not matched to subject behaviour. Failing to match subject data means that the accuracy of the implementation of the mechanism cannot be determined.

Capacity is examined in two models, but neither model provides a satisfactory capacity mechanism as defined by the theories presented in Chapter 2 (e.g. a limited number of elements that can be active at any one time; the number of elements that can be processed simultaneously). The Halford et al. model includes a number (representing mental effort) which is increased every time a production rule results in failure. Once the number exceeds a specified threshold, more complex operators can be applied to the task situation. The Buckingham and Shultz model simulates limited capacity by adding noise to the input of one of the task variables. The noise represents an increased likelihood that an incorrect value will be assigned to the variable. The two models are not included as capacity models in Table 3.3 because they use mechanisms which fail to examine capacity directly, as specified by capacity theories of development.

However, the Halford et al. model does indicate that capacity may be required to build more complex productions, and should therefore be a factor in strategy acquisition. Halford et al. state that "the question of capacity is not whether it is an alternative to any of these processes [accumulation and organisation of a knowledge base, skill acquisition, efficient encoding], but whether, and how, it interacts with them" (Halford et al., 1995, p.124). The model also suggests that capacity and experience interact, because the shift to the higher capacity is based on the experience of failure of previous strategies.

The cascade-correlation learning algorithm could be interpreted as increasing capacity because it adds a unit at specific intervals in the training cycle. The addition of the unit usually leads to an improvement in performance for the network. However, the additional units mainly serve as a transition mechanism rather than an investigation of capacity. Their function is similar to the addition of a new rule (which also enables an improvement in performance). The cascade-correlation algorithm is therefore not considered as an examination of the influence of limited capacity on performance.

The above summary of developmental mechanisms which have been implemented in computational models shows that there are few mechanisms outside of knowledge that have been implemented within a computational framework. There is an imbalance whereby several developmental mechanisms have been proposed by verbal developmental theories, but these mechanisms have never been tested within a computational model. This means it is difficult to gauge to what extent unexamined mechanisms of development may explain changes in children's behaviour.

The lack of implementation of the full range of theoretical mechanisms presents the need to examine the influence of developmental mechanisms upon behaviour. Computational modelling allows the manipulation of one developmental mechanism whilst keeping all others constant. The influence that each individual mechanism of development has on the behaviour of the model can therefore be examined in detail. Implementing several mechanisms of development within a single computational model will mean that each mechanism can be tested relative to all other mechanisms. This will provide a powerful and fair test of mechanisms of development.

3.3.2 The majority of modelling researchers are not developmental psychologists

Examining the list of modellers (in Table 3.3), it is apparent that many of the modelling community do not have a background in developmental psychology (e.g. Langley, McClelland, VanLehn). These researchers are modellers rather than developmental psychologists. It stands to reason that some of the developmental mechanisms proposed by developmental theory have been ignored, because the modellers only implement mechanisms which apply to their particular computational theory.

The majority of developmental theorists (see Table 2.1 in Chapter 2) have little background in computational modelling. This results in theories of development being

verbal (the problems of verbal theories were discussed in Chapter 2). The lack of overlapping knowledge between modelling and developmental researchers has led to the majority of research in mechanisms of development to occur in parallel: either in terms of verbal theory or in terms of computational modelling, but rarely in terms of both.

The recent resurgence of interest in using computational modelling to examine development (e.g. the Simon and Halford [1995] book) is therefore likely to have been strongly influenced by researchers that are competent in both the theorising and modelling of development (e.g. Halford, Siegler). This is a small community, but it has provided a platform by which solid theorising can be supported by computational models. However, the theorists have used computational modelling to test their own theories of development and have not tested any other theories of development. This thesis hopes to address this issue.

3.3.3 The models match subject data on only a few measures

Very few of the models attempt to match behaviour at a detailed level. This problem has been directed at both production rule based and connectionist based models (Klahr, 1995). One reason is the tasks that are modelled. For example, the balance-scale task does not permit detailed analysis of task behaviour because the detailed behaviour is not observable. The models of the balance-scale task therefore compare behaviour by matching accuracy on problems.

Selecting a task for which multiple measures of behaviour can be taken is important for modelling because it constrains the processes that can be incorporated into the model. Multiple measures also reveal more about the task behaviour and therefore enable a more detailed task analysis from which to build a model. For example, Gentner and colleagues (1995) use a relational similarity task where the only available measure is the item that was selected (from a group of three or four) that is perceived as being relationally similar to a source item. Using a single observable measure means it is very difficult to infer the underlying processes that are involved in the task. This leads to a model that is difficult to test because it can only be matched to one measure of subject behaviour.

Failing to use tasks which involve multiple measures may contribute to the success of many computational models in matching subject data when only manipulating knowledge. Using only a few measures means that low-level behaviour is ignored for which the model may not match subject data. Whilst the models may indicate that knowledge changes by

themselves are sufficient to model performance progressions (in specific domains at least, when matching a small number of measures), both developmental theory and the creators of the models (e.g. Gentner, Rattermann, Markman & Kotovsky, 1995) acknowledge that further mechanisms of development exist.

One measure of behaviour which can prove very reliable when developing a model is timing data. Timing data is ignored by *every model* that is presented in this chapter. Timing data is an important behavioural measure for several reasons:

1. It can indicate learning in a task, if the same problem is performed faster on subsequent trials.
2. It can indicate the time that is spent on each task behaviour.
3. It provides a detailed level at which to match model and subject behaviour; task processes should take the same length of time for both the model and subjects.
4. It can disambiguate strategy use.

Computational models (and verbal theories) have often been criticised because they cannot ensure that the processes used by the model are the actual processes that subjects use (e.g. Searle, 1980/1997). Increasing the number of task measures that are recorded, and including timing data as part of the measures, means there is a lot more data for the model to match. If the model is able to match subject data on a significant amount of the measures, then there can be greater confidence that the processes by which the model completes the task are correct.

In order to provide a more compelling computational model, a large amount of task measures need to be used. Current models of development have failed to match subject data on multiple measures of task behaviour. This is problematic because it increases the likelihood that the model is using different processes to subjects when completing the task. Using multiple measures is particularly important when analysing the change in behaviour that modifications to the model produce, because modifications may only change a subset of the models task behaviour. The computational model that will be developed in this thesis will need to satisfy the requirement of matching subject data on multiple measures.

3.3.4 Problems operationalising the mechanisms proposed by developmental theory

The idea of implementing several mechanisms of development, many of which exist in only a verbal format, highlights a specific problem: the mechanism must be interpreted correctly and implemented correctly within a computational framework. This is a problem to be aware of but it is also one which may be difficult to address. If the details of a mechanism are found to be vague, then the modeller is forced to make assumptions about the mechanism when implementing it within a computational framework. This may not be consistent with what the theorist meant in the verbal explanation of the mechanism.

The balance-scale models by McClelland and Jenkins, and by Shultz, Mareschal, and Schmidt, are both cited as including developmental mechanisms that map onto the accommodation and assimilation developmental mechanisms proposed by Piaget. Accommodation and assimilation are treated slightly differently by each model. This highlights the problem of operationalising theoretical mechanisms within two similar computational frameworks.

Plunkett and Sinha (1992) claim that the McClelland and Jenkins model has mechanisms that can map onto Piaget's concepts of assimilation and accommodation. Two assumptions are critical to the model's behaviour: the separation of the weight and distance in the hidden units (the structural assumption), and the bias for weight in the input data (the input assumption). Assimilation arises from the structural assumption; the input data has to be assimilated into the network, which is achieved by the change in weights between units for each individual input pattern. Accommodation arises from the input assumption; repetitions of the input data cause the network's representation to change, enabling it to co-ordinate the weight and distance dimensions.

The cascade-correlation models are also said to map onto the concepts of assimilation and accommodation (Shultz, Schmidt, Buckingham & Mareschal, 1995). Assimilation is viewed in the same way: the changing of weights between the units for each input pattern. Accommodation is viewed as changing the structure of the network to fit in with new information. This corresponds to the phases in the training where hidden units are recruited. As with Piaget, this new knowledge builds upon previous existing knowledge.

The two different operationalisations of Piaget's theory provide a classic example of how the theory can be interpreted differently because of its vagueness. Two very similar network architectures explain the notion of accommodation differently. The problem does not lie with the operationalisation of the theory, it lies with the vagueness of the theory. Lack of specification has led to different interpretations of the theory.

3.4 Summary

Current models of development have not implemented the full range of developmental mechanisms that are specified by developmental theories. The developmental mechanisms that have been ignored within models have therefore never been tested, and their worth is unknown within a computational framework. In order to test mechanisms of development, the model must be matched to the subject data on a variety of measures, because the mechanisms may only affect a subset of the models behaviour.

This thesis proposes to examine mechanisms of development by creating a computational model in which the mechanisms of development described in Chapter 2 can be implemented. The resulting model will enable each theoretical mechanism of development to be tested. Each mechanism can be implemented independently of other mechanisms in order to see what influence it has on the behaviour of the model. Examining the full range of developmental mechanisms within a single model will help to extend the detail of these mechanisms, and will help to provide insights into the factors that transition mechanisms must take into account.

This chapter has covered several models of developmental tasks. It showed that the mechanisms of development used by the models do not cover the array of developmental mechanisms that theories of development propose. A need to examine all of the proposed mechanisms of development within a computational framework was put forward. The next chapter details a developmental task in which multiple measures of behaviour can be observed. The task will provide the basis for a computational model.

4. A task in which to study child development

The preceding chapters have argued for testing different mechanisms of development within a single computational model. The model must be capable of matching subject data on multiple measures because this provides model–subject comparisons at a low-level. Multiple measures also show which measures are influenced by specific developmental mechanisms and which are not. A task is therefore required which is tractable to model computationally, and which has observable behaviour where multiple measures can be taken. This chapter details a developmental task which fulfils these criteria.

The chapter explains the developmental task, together with the terminology that is used to describe various aspects of the task. Previous studies that have used this task are described which show that children's performance on the task improves with age. Both adult's and seven year old children's performance on the task are covered in detail. The adult behaviour will be used in Chapter 5 to develop a model of task behaviour. The seven year old behaviour will be used in Chapters 7 and 8 as a comparison to the behaviour of the model when it is modified to simulate development.

4.1 The Tower task

The Tower task (Wood & Middleton, 1975) is a problem solving puzzle in which a pyramid (shown in Figure 4a) must be assembled from a set of 21 wooden blocks . There are six layers to the pyramid; the lower five consist of four blocks each, with a single block as the top layer. The blocks which comprise each layer are all of the same size, but the size of blocks changes uniformly across layers. The blocks in the lower layers all share the same characteristics (shown in Figure 4b), differing only in size.

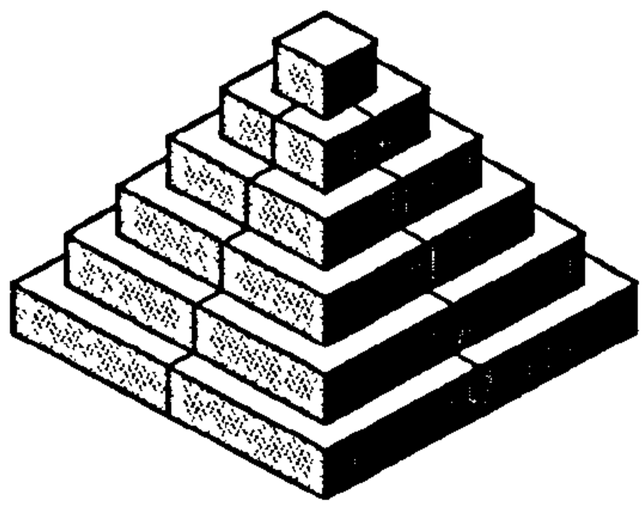


Figure 4a: The final assembly of the 21 blocks that comprise the pyramid puzzle.

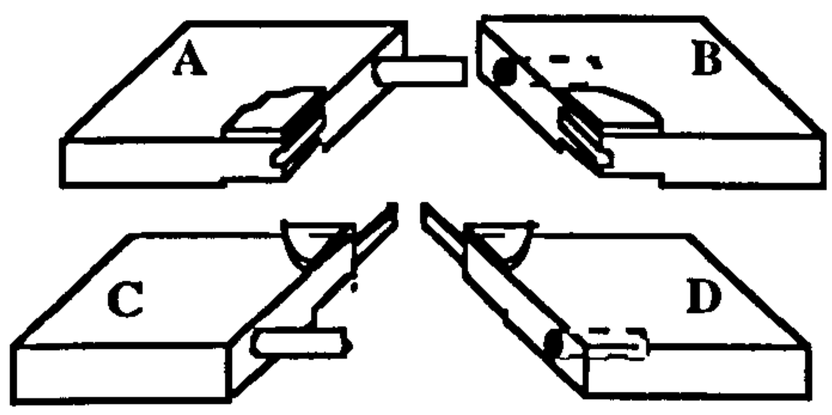


Figure 4b: The four blocks that make up each of the lower five layers in the pyramid puzzle.

4.1.1 Characteristics and terminology in the Tower task

The single block which comprises the top layer has only one salient feature: a circular depression. Every other block in the Tower task has four salient features: a quarter circle depression (except for blocks in the bottom layer); a quarter circle elevation; a peg or a hole; a halfpeg or a halfhole. Each block can be assigned a unique label based on its size and its features, although quarter circles are ignored (because every block has them). For example, Block A in Figure 4b is referred to as a halfhole-peg block, and block B a halfhole-hole block. The size of blocks is referred to by number (six being the largest and one being the smallest), so the largest C block is Size6-halfpeg-peg.

The block features allow the Task to have several interesting characteristics. First, every peg and half-peg can fit into every hole. Second, the position of pegs and holes from the edge and bottom of each block is the same, such that placing the peg of one block into the hole of a different sized block can result in a construction which is "flush" on its outer edge. Third, each layer is formed by putting together correctly the four blocks comprising a layer (excluding the single block top layer), such that the quarter circle elevations on each block form a circular elevation, and the quarter circle depressions form a circular depression. The diameters of the circles are the same for every layer, permitting the stacking of layers in any order of size. Fourth, the six layers all differ in size by the same magnitude. For example, the difference in size between layers two and three is the same as that between layers five and six (this is shown in Figure 4a).

The block characteristics allow three general strategies in forming a layer. First, fitting the peg of block A into the hole of block B brings the two halfholes together to form a pair having a hole (a hole-pair). Similarly, fitting block C and block D together forms a pair with a peg (a peg-pair). A layer is then formed by fitting the peg of the peg-pair into the hole of the hole-pair. Second, fitting the halfpeg of block C into the halfhole of block A forms a pair having two pegs (a two-peg pair). Similarly, fitting block D and block B together forms a pair having two holes (a two-hole pair). The two-peg and two-hole pairs can be fit together to form a layer. Third, an initial pair can be constructed, and then a block added to the pair (a three-block construction), and another block then added to this to form a layer. Several different three-block constructions can be formed (because each of the different layer blocks can be remaining), and so the identification name of the three-block is suffixed with the remaining block's label. For example, a three-block formed from a peg-pair and a halfhole-hole block is called a three-block-halfhole-peg. Although there are different strategies that can be used to construct a layer, they are all equally efficient because each strategy, if carried out successfully, requires three construction operations.

A correct construction in the Tower task is one which furthers progress in completion of the Tower. This includes all the constructions produced from the strategies mentioned above (assuming the constructions comprise blocks of the same size), and the appropriate stacking of layers. Error-free performance involves twenty correct constructions; three for the production of each of the five layers, and five for stacking layers.

The differences in size of blocks, and the variety of features that blocks have, enable a variety of incorrect constructions to be made. How an incorrect construction is produced can help give insights as to what task knowledge is known. For example, if a subject always produces incorrect constructions which are flush on their outer edges, then this implies that some knowledge of the appearance of correct constructions is known.

There are four general attributes that can be shared between incorrect and correct constructions. For each, an example of an incorrect construction having the attribute is given.

1. Flush. This is when the outer edges of a construction are all flat. Figure 4c shows both a construction with flush outer edges (which also has the peg of one block attached to the hole of the other), and a construction with jagged outer edges. The flush attribute introduced here does not involve the features of blocks, but can be an attribute of a construction.

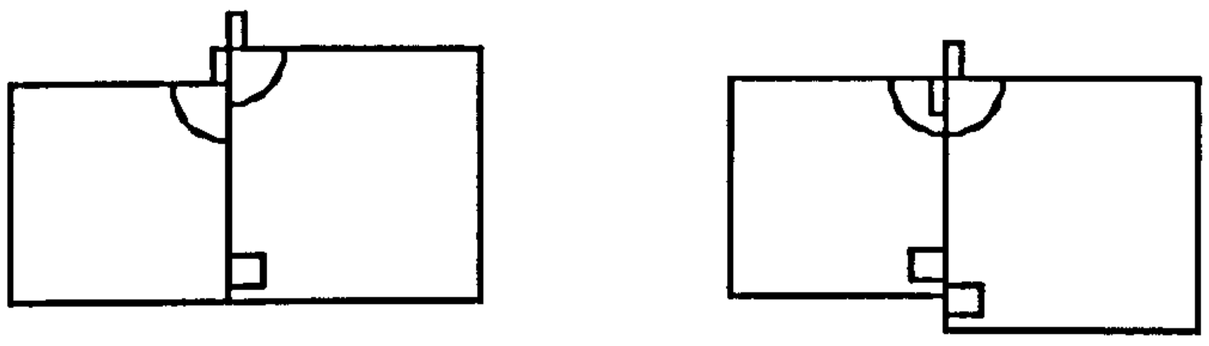


Figure 4c: A construction with flush outer edges (left) and one with jagged outer edges (right).

2. Quarter circles aligned. The quarter circles of each block are aligned to form a semi-circle (or three quarter circle, depending on how many blocks are in the construction). An example is the incorrect alignment of the halfhole-hole and halfhole-peg blocks, as shown in Figure 4d. This construction has the appearance of a correct construction because it is flush on its edges and it has a semi-circle in its centre.

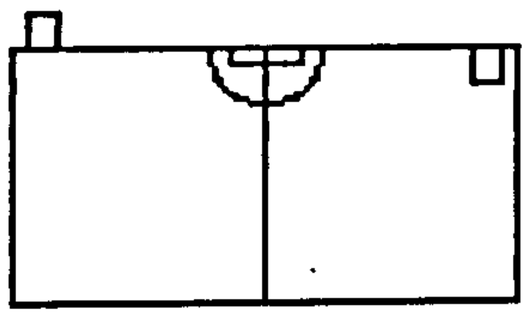


Figure 4d: A construction with quarter circles aligned.

3. Peg in hole. When a peg of one block is placed in the hole of another. For blocks of the same size, this results in a jagged-edged construction, as shown in Figure 4e.

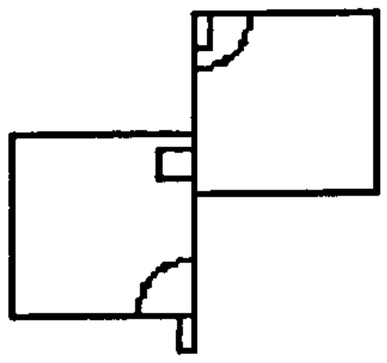


Figure 4e: A construction with the peg of one block attached to the hole of another.

4. Halfpeg in halfhole. When a halfpeg of one block is placed in the halfhole of another. For blocks of the same size, this results in a jagged-edged construction (as shown in Figure 4f) in which the halfpeg and halfhole do not fit well together. The halfpeg and halfhole do not fit well because the curved edges of each do not face each other.

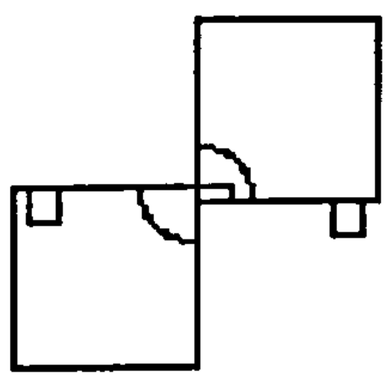


Figure 4f: A construction with the halfpeg of one block attached to the halfhole of another.

For any two blocks of the same size, an incorrect construction cannot involve more than one of the attributes 2-4, because it would then be a correct construction. Co-ordinating more than one feature at a time on the Tower task means a correct construction has to be made (assuming blocks are of the same size and are appropriately matched).

The blocks that produce each layer share the same characteristics; learning should occur within the task such that subsequent layers take less time to construct than previous layers. Within-task learning is useful to examine because it presents a slightly different view from Simon (1962). The behaviour at each stage does not remain static, it still undergoes modification. This viewpoint is supported by Klahr (1995).

4.1.2 Probability of success when fitting features

The block features can either be attached to each other, or aligned to each other. The appropriate feature combinations that can attach, or fit into each other, are pegs and holes, halfpegs and halfholes, and quarter circle elevations and depressions. Appropriate feature combinations that can align to each other are halfholes, halfpegs, and quarter circles. A correct construction is guaranteed for blocks of the same size that are produced by attaching or aligning two or more pairs of features.

Assuming any two blocks of the same size are selected, and both of the blocks are the correct way up, then there are different probabilities of success for fitting features that can attach, or fitting features that can align. Table 4.1 shows the probability of each type of possible fit. Note that quarter circle indents fitting into quarter circle depressions are ignored because this is a stacking operation (it is not an appropriate operation for blocks of the same size).

Table 4.1: Probability of success when fitting features of two blocks of the same size (which are both the correct way up). *Some of the ways of aligning quarter circles do not result in an incorrect construction because the protruding features of some blocks (e.g. a peg) mean their quarter circles can only align in one way.

Fit type	Fitting features	Probability of success
Attach	Peg in hole	50% (2 out of 4)
Attach	Halfpeg in halfhole	50% (2 out of 4)
Align	Halfpegs	100% (1 out of 1)
Align	Halfholes	100% (1 out of 1)
Align	Quarter circles	33%* (4 out of 12)

The probability of aligning quarter circles successfully is the lowest because quarter circles appear on every block, and can align in two different ways for each pair of blocks. This should therefore yield a 50% success rate, but there are two sets of paired blocks which cannot be fit together successfully (the halfpeg-peg block with the halfhole-hole block, and the halfpeg-hole block with the halfhole-peg block). The success rate is therefore lower than for any other combination of features.

Quarter circles can be aligned to make constructions that can *appear* to be correct (see Figure 4d). However, correct constructions have either a peg (formed from two halfpegs), a hole (formed from two halfholes), two pegs, or two holes. The incorrectly aligned quarter circle constructions will not.

4.1.3 Why use this task to study development?

The variety of block features and the need to co-ordinate them means that the puzzle is complex for children aged between three and eight. There are several task aspects which the children find difficult:

- There are a variety of ways in which both correct and incorrect constructions (errors) can be made. Correct constructions must be recognised from incorrect constructions.

- Selection of blocks must take into account three criteria for assembly: size, nature of connecting pieces (e.g. whether they have pegs, holes, halfpegs, halfholes, and so), and the relative orientations of blocks and constructions (because they will need to be manoeuvred to face connecting blocks the correct way up and at the correct orientation).
- Layers must be stacked in a size ordered sequence, because the block characteristics permit the piling of layers not seriated for size.

The task complexities listed above mean there is a wide range of task behaviour, where task performance improves with age. Three year old children are complete novices who can hardly be taught the task, whilst eight year old children are relative experts who can teach themselves. Older children accomplish more correct operations, produce less errors, and take less time than their younger counterparts (Murphy & Wood, 1981; Wood & Middleton, 1975). This allows the study of children's problem solving behaviours across ages on the Tower task.

As the Tower task is a physical problem solving puzzle, a detailed analysis of task behaviour is possible via videotape. Many strategies are readily visible, reducing the need for the experimenter to infer mental structures and strategies. This enables a more accurate computational model to be created.

4.2 Previous studies using the Tower task

Studies that have used this task have been mainly concerned with assessing the various tutoring strategies that can be employed to teach children how to assemble the pyramid. When asking mothers to teach their children the task, Wood and Middleton (1975) found that four general tutoring strategies emerge. Some mothers simply demonstrated the task to their child; some gave only verbal descriptions of how to build the pyramid and did not even touch the blocks; others swung between these two; others gave their children "contingent" help based on their achievement thus far, in that if the child was successful they gave less help next time, and if unsuccessful they gave more help.

Wood, Wood, and Middleton (1978) then examined each strategy with three to four year old children, discovering that contingent tutoring easily out-performed the other three methods in a post-test. Children who had been exposed to contingent tutoring averaged 15.0 correct constructions, whereas the best of the other tutoring strategies averaged 6.2 correct constructions.

Wood, Bruner, and Ross (1976), using a strategy much like the contingent one, found that there is a progression in performance with age on several task measures, such as the number of constructions per intervention, and how successful each intervention by the tutor was. One interesting finding of this study was that three year olds could recognise a correct construction without actually being able to produce one. The three year olds disassembled almost as many correct constructions as incorrect ones, but reassembled 66% of the correct ones as compared to 14% of the incorrect ones. These percentages matched those of four year olds (although four year olds were less inclined to disassemble correct constructions).

Murphy and Wood (1981; 1982) compared task performance when children (aged between four and eight) were given three types of instructional aid: nine pictures showing the stages to completion of the Tower; a silent film of a demonstration of the Tower being completed from start to finish; and a verbal description of the completed Tower. Children in the picture and film groups performed significantly better than the "non-instruction" group in terms of how many completed the Tower, the number of correct constructions, the number of errors made, and the time taken per correct construction.

For all of the ages of children in the Murphy and Wood studies (5, 6, and 8 years; the 4 year olds in the non-instruction group found the task too distressing), the non-instruction group performed worse than the others with respect to efficiency (number of correct constructions divided by total number of constructions) and in terms of errors made. Although the non-instruction group fared the worst for the number of correct constructions for each age band, no significant difference between groups was found.

The non-instruction group in the Murphy and Wood studies are of interest because this enables the assessment of ability across ages when children do not receive help (they are only told the final appearance of the Tower). Comparisons between ages reveals that there is a performance progression at each age band (5, 6 and 8 years), elders performing better in terms of less error, more correct constructions, less time to complete the task and less time per correct construction. However, there are no significant differences between ages for most of these measures.

4.3 Modelling the task

Producing a model of the Tower task involves a detailed analysis of subject's task behaviour. It must therefore be decided which subjects to use in producing the first model of the task.

4.3.1 Where to start

There are two natural ways to create a series of developmental models. One way is to model a lower performance level (that of children) and modify the model to fit higher performance levels. The other way is to begin at the highest performance level (that of adults), and then modify the model to fit lower performance levels. Young children's behaviour on the task can be considered chaotic. They make a lot of errors and have great difficulty in completing the task. For these reasons the highest performance level is chosen as a start point. Another advantage of using adults over children is that verbal protocols can easily be taken, which helps elicit what knowledge and processes subjects may be using whilst constructing the Tower.

Klahr (1995) believes that the "dumbing down" of adult models is not the way to proceed for examining *transition* mechanisms. However, it would seem to be the obvious place to start for examining *developmental* mechanisms and how they influence behaviour. This is because a precise specification of behaviour can be achieved from studying adult performance, which in turn means a detailed model can be created. Data will be able to be matched at a low-level. Few models have matched children's data at a detailed level (see Chapter 3).

4.3.2 Where to stop

This thesis proposes the methodology of modifying architectures to examine children's development. The focus is on showing that modifying architectures enables the examination of the different methods of development that are proposed by developmental theories. The goal of the thesis is not to present a series of models which are able to fit children's data at different ages. Given that the starting point is adults on the task, the point at which to stop is the next immediate point where behaviour on the task changes significantly. The data from seven year old children fulfils this criteria. The data from both adults and seven year old children will be analysed.

4.4 Analyses of adult and seven year old behaviour on the Tower task

The data from adults and seven year old children will be compared, rather than presenting each set of data individually. The data from adults was obtained in a previously unreported study; the data from seven year old's involved re-analysis of a previous study. The details of each of these studies will be given, followed by the results from each (described as a comparison between the two sets of results). Before detailing each experiment, the coding scheme will be covered, to provide the reader with some knowledge of the type of task behaviour being analysed.

4.4.1 Scheme for coding the task behaviour

There are three distinct types of overt behaviour that occur when the Tower task is being completed: Constructing and disassembling; manipulating blocks (e.g. rotating them); and selecting blocks. The constructing and disassembling behaviour indicates task knowledge and goals. Manipulating blocks and selecting blocks are difficult to determine goals for. For example, selecting a Size6-Halfpeg-Peg block could mean the subject simply selected a random block of the largest size, a random block which had a specific feature such as a peg, or a combination of these. To arrive at any conclusions as to why subjects manipulated blocks in a specific way, or selected specific blocks, is therefore subjective. Only the constructing and disassembling behaviour will be examined (aside from specific tutoring measures, the previous studies only examined construction behaviour). The construction behaviour will be analysed in much finer detail than in the previous studies.

A brief summary of the codes is shown in Table 4.2. The full coding scheme is detailed in Appendix A, and an example of a full task coding sheet (the coding for subject 1 of the seven year old's) is shown in Appendix B. Each code takes as arguments the blocks or constructions that are involved in the coded behaviour (the block and construction labels are detailed in the *Characteristics and terminology in the Tower task* section).

Table 4.2: Summary of the construction behaviour codes for the Tower task.

Code	Description	Example
Correct	A construction which furthers progress in the Tower. When a stacking operation is performed, the code is suffixed with "qc-d-in-qc-e" (quarter circle depression in quarter circle elevation).	Correct(Size6Halfpeg-Peg,Size6Halfpeg-Hole)
Incorrect	A construction which does not conform to any of the correct constructions detailed in *Characteristics and terminology in the Tower task*. The incorrect code is suffixed by the task appropriate features that exist for the construction (also detailed in *Characteristics and terminology in the Tower task*).	Incorrect/flush/qc-aligned(Size6Halfhole-Hole,Size6Halfhole-Peg)
Disassembles	Disassembling a construction to leave all of the constituent blocks unconnected. The code is suffixed with "correct" or "incorrect" based on whether the construction being disassembled was correct or incorrect.	Disassembles-correct(Size6HolePair)
Removes	Disassembling a construction to leave some of the constituent blocks connected (i.e. leaving either two constituent constructions, or one constituent construction and a single block). The code has three arguments, the first two are the blocks/constructions that remain after the disassembly, the third is the construction that was disassembled. The code is suffixed with "correct" or "incorrect" based on whether the construction being disassembled was correct or incorrect.	Removes-incorrect(Size6Halfhole-Hole,Size6PegPair, Incorrect[Size6Halfhole-Hole,Size6PegPair)

The correct and incorrect codes can be prefixed by "Considers" when blocks or constructions are aligned ready to fit, but the subject gazes at them for two seconds or more (a physical fit between the blocks or constructions may or may not take place afterwards). The behaviour suggests that the subjects are performing some form of mental operation which decides whether the fit should be carried out or not.

When the behaviour that is appropriate to a code occurs, the code is recorded along with the time (from the beginning of the task, to the nearest second). The coding method enables a variety of timing information to be obtained, as well as all of the construction information.

4.4.2 Study from which the data from adults is obtained

The full method for the adult Tower task experiment is given in Appendix C. The relevant aspects of the experiment are detailed here. Five adult subjects participated in the experiment, all of whom had never seen the Tower task before. The subjects initially completed a simple child's jigsaw, whilst talking aloud, in order to get accustomed to giving verbal protocols (as suggested by Ericsson & Simon, 1984). The subjects were then presented with a picture of the completed tower (the same as shown in Figure 4a) together with written instructions stating that the goal of the task was to build the tower from a set of blocks, whilst talking aloud. The subjects were then presented with the blocks in a fixed pattern, and their task performance and verbal protocols were recorded on videotape for task analysis.

4.4.3 Study from which the data from seven year old's is obtained

All of the previous Tower task studies have examined children's performance with instruction. The analyses of the data focused on how well the tutoring experience worked. The analyses did not require a fine-grained analysis of the subjects' task behaviour. The development of the model will result in a variety of measures to compare to subjects. Comparing the model's behaviour with the behaviour of seven year old children will mean a more detailed analysis of the data from seven year olds being carried out.

A more recent study of children's performance on the Tower task was carried out (Reichgelt, Shadbolt, Paskiewicz, Wood & Wood, 1993) using a computer based intelligent tutoring system (EXPLAIN; Wood, Shadbolt, Reichgelt, Wood & Paskiewicz, 1992) . The study included videotaped behaviour of sixteen seven year old children completing the Tower task. The behaviour of five of the sixteen children is reported here.

The five children were all contingently tutored to complete the Tower, and then performed a post-test where the children were asked to complete the Tower without help. The re-analyses here are based on the children's post-test behaviour. The design of the experiment can be found in Appendix D.

Using tutored seven year olds could be problematic, because their task knowledge may be more advanced than standard seven year old children. The study from which the seven year old data is taken states that contingent tutoring is not needed from the age of six and upwards (Reichgelt et al., 1993, p.244).

A re-analysis of the tutored and post-test performances of the seven year old's reinforces the view that contingent tutoring does not significantly add to seven year old's behaviour. The seven year old's rarely required help during tutoring (averaging 3 interventions by the tutor), suggesting they can complete the task without any tutoring. The time taken to complete the task during computer-based tutoring is significantly longer than for the post-test (447.4 s versus 134.1 s; $t(8)=4.43$, $p<0.01$). This is expected because the children have to wait for the computer to issue commands; when the children progress quickly the tutor takes some time to catch up. A better comparison is on construction attempts. There is no difference between the number of construction attempts made in pre-test and post-test (28.4 versus 27.6; $t(8)=0.26$, $p>0.05$). The contingent computer-based tutoring does not add a significant amount to the knowledge base of seven year old children on the Tower task.

4.4.4 Results

The results of the adult experiment and re-analysis of the seven year old data will be detailed in three sections: overall measures, strategy measures, and within-task measures. The overall measures indicate general task behaviour, such as the time taken to build the Tower, and the number of constructions made in building the Tower. Strategy measures indicate the types of correct and incorrect constructions that are made. Within-task measures indicate if there is any learning taking place whilst subjects build the Tower. This is necessary because the task involves the building of five layers where the blocks in each layer share the same characteristics. This means there is within-task learning: each subsequent layer generally takes less time to construct.

4.4.4.1 Overall measures

A variety of overall measures exist. The most important (because they define the task and influence scores on other overall measures) are the number of construction attempts and

the time taken in completing the Tower. After detailing these two measures, further measures are examined: the breakdown of constructions; remembering previous fit attempts; and the timings between constructions.

Construction attempts and time taken

Table 4.3 shows the total number of construction attempts and the total time taken in constructing the tower, for the adults and the seven year old children. There is a significant difference between the construction attempts scores ($t(8)=2.34$, $p<0.05$); the difference between the time taken scores is not quite significant ($t(8)=2.27$, $p>0.05$).

Table 4.3: Construction attempts and time taken in completing the Tower task, for adults and seven year old children. Standard deviations are shown in parentheses.

	Adults	Seven year old children
Construction attempts	22.8 (2.9)	27.6 (3.5)
Time taken	126.6 s (34.0)	174.0 s (32.1)

It could be that the extra time taken by seven year olds is because they make more constructions. The average time per construction for adults is 5.2 s; for seven year olds the average is 6.3 s. There is no reliable difference between the two ($t(8)=1.70$, $p>0.05$), but a 1.1 s margin per construction indicates that the extra time taken by seven year olds cannot be solely consumed by their increased number of construction attempts.

Breakdown of constructions

The construction attempts can be broken down further (see Figure 4g for adults, and Figure 4h for seven year old children). An average of more than 20 correct constructions (the optimal amount) means that some correct constructions get erroneously disassembled. The number of correct constructions produced is not of major importance (the adults and seven year olds do not differ on this score anyway), but the number of incorrect constructions produced is, because this indicates the errors made on the task. Two measures reveal differences between adults and seven year old children: the incorrect constructions involving same size blocks, and the incorrect constructions involving different size blocks.

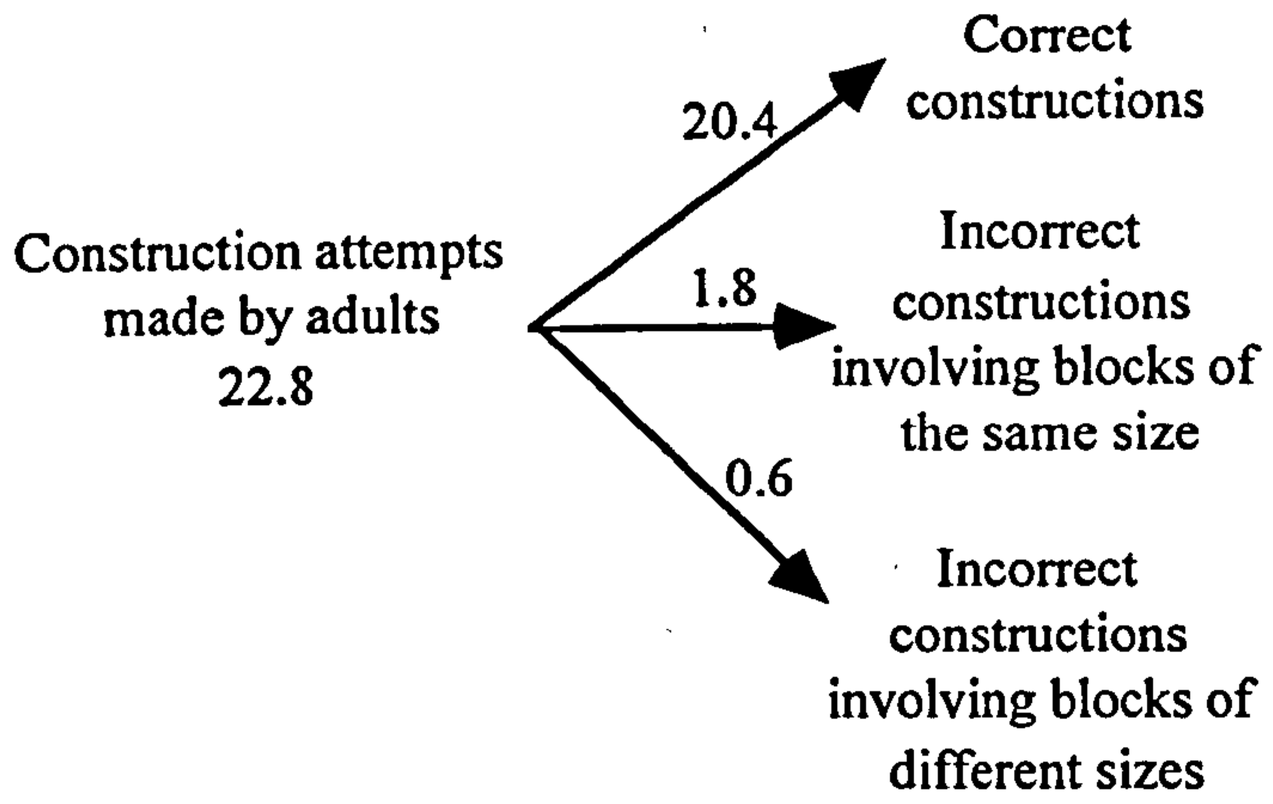


Figure 4g: Breakdown of the construction attempts made by adults.

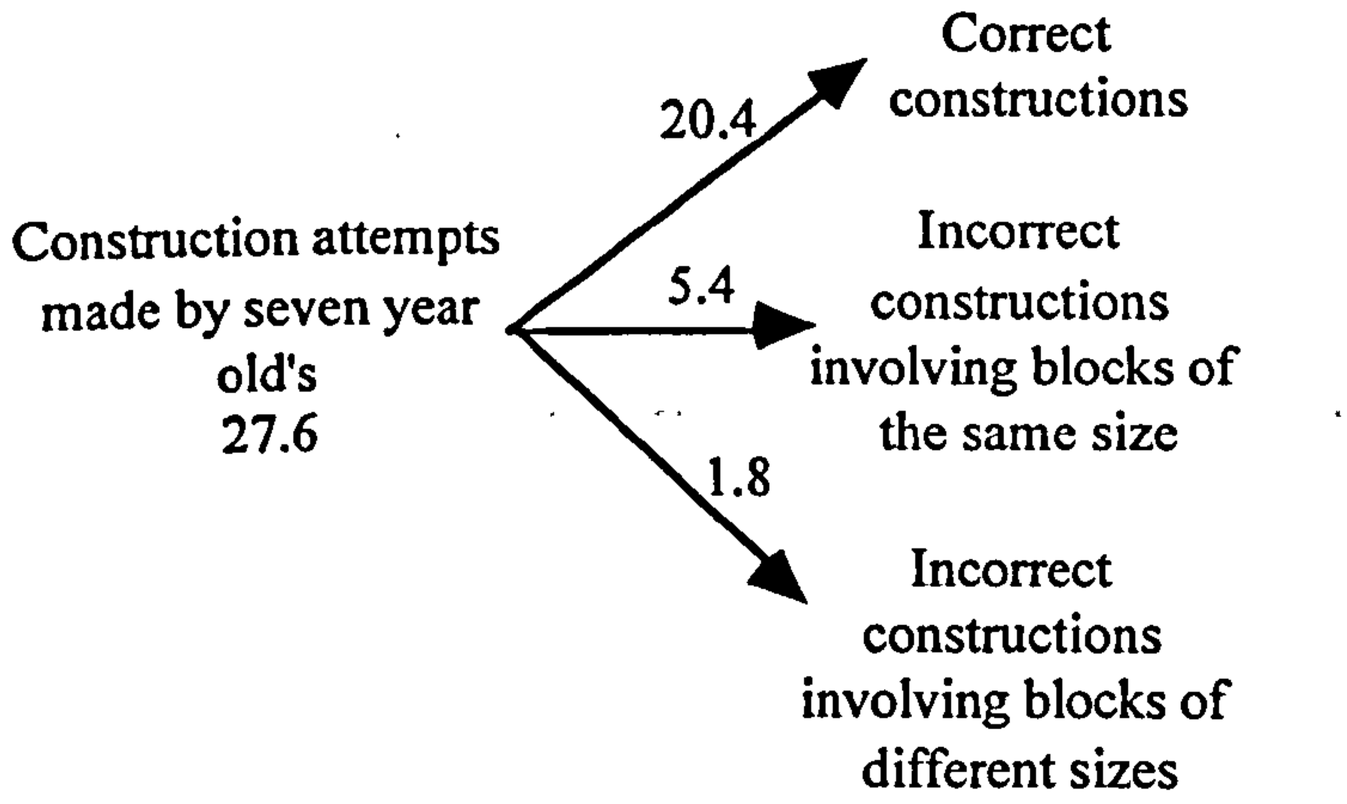


Figure 4h: Breakdown of the construction attempts made by seven year old children.

There are no reliable differences between adults and seven year olds for the number of correct constructions, and the number of incorrect constructions involving blocks of different sizes. A significant difference exists for incorrect constructions involving blocks of the same size ($t(8)=3.22$, $p<0.05$).

Remembering previous fit attempts

After disassembling a construction involving two blocks, a decision between three options has to be made before attempting the next construction. First, the two blocks can be put aside, and completely different blocks picked up. Second, one of the blocks can be retained and one completely different block picked up. Third, the same two blocks can be tried again. The third option involves some memory of how the blocks were fit together on the previous attempt. Capacity theories suggest that memory may influence performance (see Chapter 2), and so the third option will be examined in more detail. The main thing to remember about previous fit attempts is the blocks and the features of the

blocks that were involved in the fit. Two measures can help examine these: the number of times the same blocks were fitted together in different ways indicates memory for the blocks and the specific way they were fit previously; the number of times the same blocks are fit together in the same way indicates a lack of memory for remembering how blocks were fit together previously. Table 4.4 shows how the adults and seven year old children score on the two measures.

Table 4.4: Ways in which the same two blocks are fitted together, for adults and seven year olds. Standard deviations are shown in parentheses.

	Adults	Seven year old children
Average number of times the same blocks are fit together in the same way again	0	1.0 (0.4)
Average number of times the same blocks are fit together in different ways	1.3 (0.6)	1.2 (0.5)

Considering that the adults average 2.4 incorrect constructions, and the seven year olds average 7.2, it is not surprising that the scores are relatively low for the measures in Table 4.4. For this reason the measures do not reliably reveal anything about capacity for the Tower task. There are no significant differences between adults and seven year olds on either measure.

Timings between constructions

The average time to produce each correct construction indicates the efficiency with which subjects complete the task. This measure is not simply the time taken divided by 20 (the optimal number of correct constructions), because subjects can disassemble correct constructions. The timing between correct constructions therefore also provides an indication of how often correct constructions get disassembled. Adults average 6.0 s to produce a correct construction. Seven year old children take 8.5 s. There is a significant difference between the two ($t(204)=1.98$, $p<0.05$). Adults are faster to produce correct constructions than are seven year old children.

The time between producing an incorrect construction and disassembling it shows the efficiency by which errors are corrected. This measure interacts somewhat with the type of incorrect construction made, because some incorrect constructions are easier than

others to identify as being incorrect. Adults average 5.0 s to disassemble an incorrect construction, and seven year old children average 2.9 s. The adults take longer because their incorrect constructions have more task appropriate features (see earlier), and therefore look more like correct constructions. There is no reliable difference between the two scores.

Summary of the overall measures

There were eight overall measures of behaviour presented. There are clear differences between adults and seven year olds on most of the overall measures of construction behaviour. Many of the differences are not significant because of the few subjects involved (analysis of videotaped behaviour is labour intensive). However, adults clearly take less time and produce less errors than seven year old children. The eight overall measures will provide an important area for the model to match the subject data for, because the measures provide a general overview of behaviour on the task.

4.4.4.2 Strategy measures

There are two types of construction that can be made when building the Tower: one which is incorrect, and one which is correct. The type of incorrect construction can provide some indication of what features subjects attend to when making constructions. The type of correct construction shows the strategies used when making a layer.

Type of incorrect construction

The type of incorrect construction needs examining carefully, because the constituent blocks in an incorrect construction may have features that are fit together correctly. This may indicate task knowledge regarding these features. Figure 4i shows the task appropriate features of incorrect constructions, for adults and seven year old children. There are no reliable differences between any of the scores (because of noise), although the magnitude of the scores is visibly different.

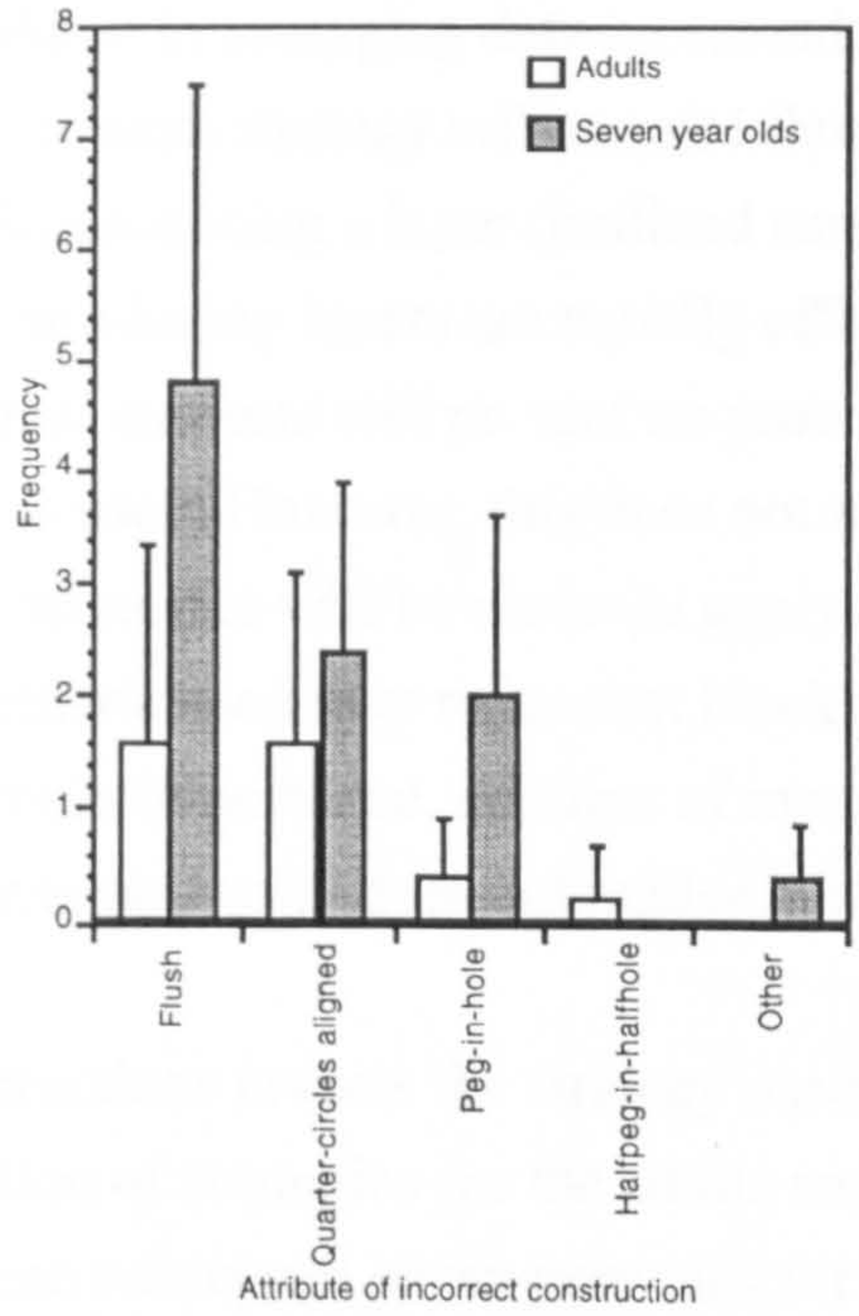


Figure 4i: Frequency distributions for attributes of incorrect constructions, for adults and seven year olds. The error bars show standard deviations.

A more reliable measure to compare adults and seven year old's on is the *number* of features that are task appropriate per construction, as this is less noisy. Table 4.5 shows the scores on this measure for adults and seven year old children. There is a reliable difference between the two ($t(44)=2.30$, $p<0.05$).

Table 4.5: Number of task appropriate features that are involved in an incorrect construction. Standard deviations are shown in parentheses.

	Adults	Seven year old children
Number of task appropriate features	1.9 (1.0)	1.3 (0.6)

Type of correct construction (layer strategies)

Each layer in the Tower task consists of blocks which have the same characteristics. This is useful because it means that for every layer, the external problem representation is the same. There is no single strategy that can be better for producing one layer than it is for another layer. All strategies are equally efficient across all layers. This means there is no problem in averaging the strategy used for constructing each layer.

However, there may be a problem in averaging data across subjects (Siegler, 1987) . When constructing each layer, any optimal strategy will require three construction attempts. Each of the three strategies for producing a layer (outlined earlier) require three correct operations. All strategies for producing layers are equally efficient. This suggests that even averaging strategies across subjects will present no problems, because it doesn't really matter which strategy is used. However, this does not account for prior knowledge (which may mean that some strategies will be easier to apply than others), and the time spent searching (the search criteria used may mean that blocks are easier to find for some strategies than others). The main importance, in terms of modelling, is that every strategy is covered. Averaging data over strategies does not matter in this endeavour.

Analysis of the correct constructions reveals the strategy used in completing each layer. Figure 4j shows the distribution of strategies for the adults and seven year olds. There are no reliable differences between adults and seven year olds for any strategy. The seven year olds never construct layers using the two-peg/two-hole strategy. The seven year olds carry out the three block strategy an average of three out of the five layers. Over half of the three block strategies begin with either a two-peg or two-hole pair (an average of 1.6 of the 3.0 layers produced by a three block strategy begin with a two-peg or two-hole pair). It is not that seven year olds have difficulty constructing two-peg or two-hole pairs.

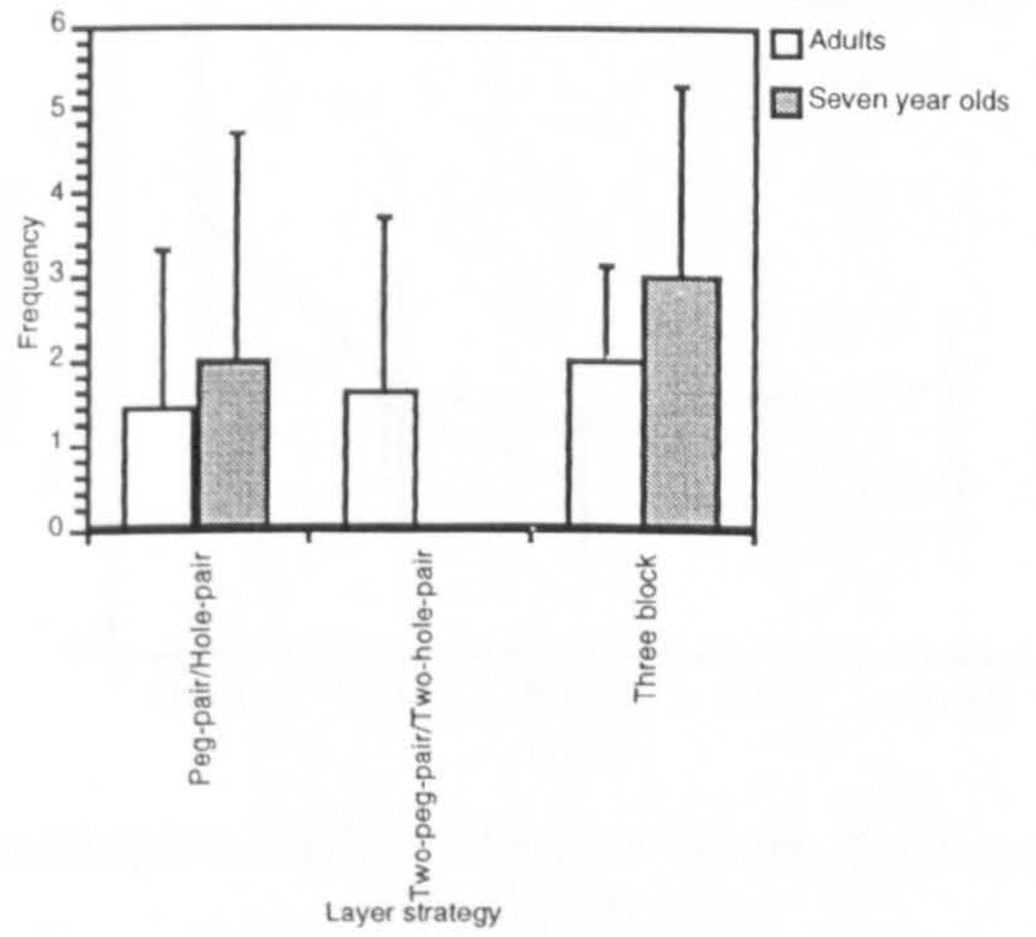


Figure 4j: Frequency of layer strategies for the adults and seven year old children. The error bars show standard deviations.

4.4.4.3 Within-task measures

Learning occurs throughout the task because the layers of the Tower share the same block characteristics. Subjects should be faster at constructing subsequent layers in the task as

they become more familiar with the block and construction characteristics. Ideally, the eight overall measures (outlined earlier) would be examined on a layer-by-layer basis, but as most have very low scores, only times and construction attempts will be examined here. Figure 4k shows the construction attempts taken in producing each layer, and Figure 4l shows the time taken to construct each layer. Adult and seven year old construction attempts do not correlate (r=0.17) because the adult construction attempts curve is flat, but the times correlate well (r=0.88). For the error bars, seven year olds are to the left and adult are to the right.

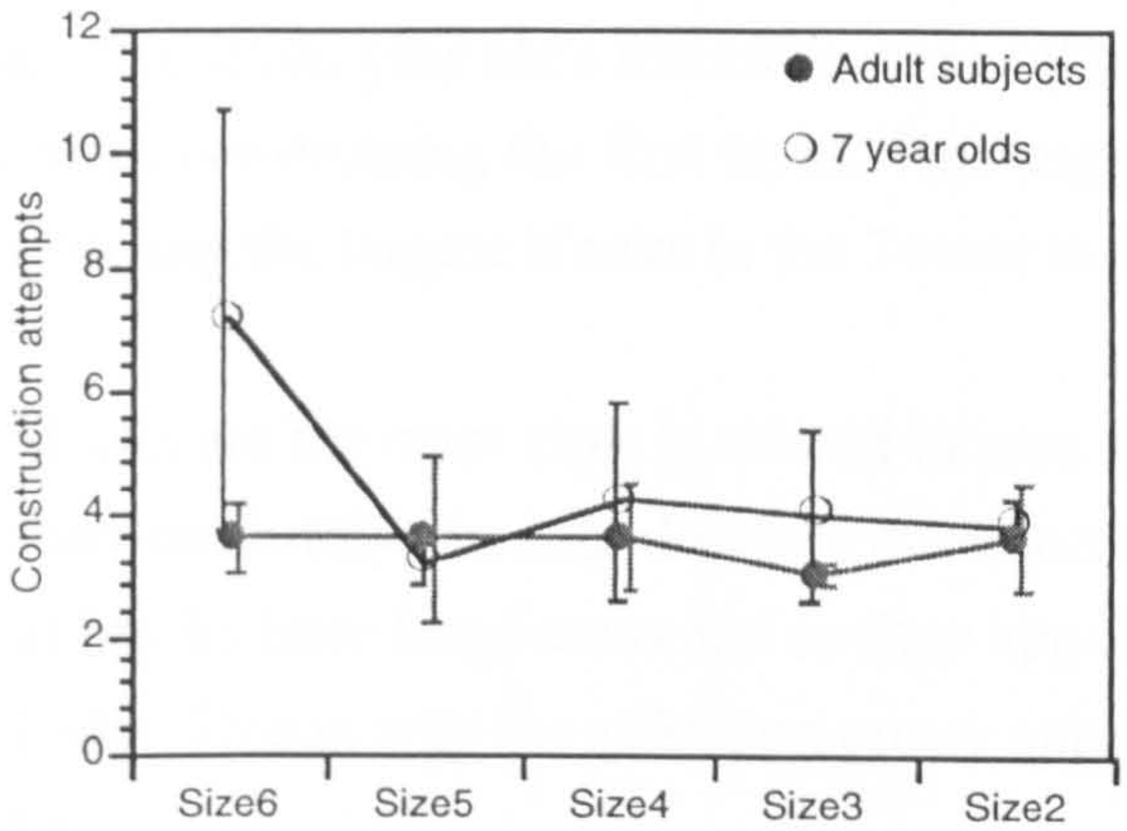


Figure 4k: Constructions made in completing each of the five layers, for the adult and the seven year old subjects.

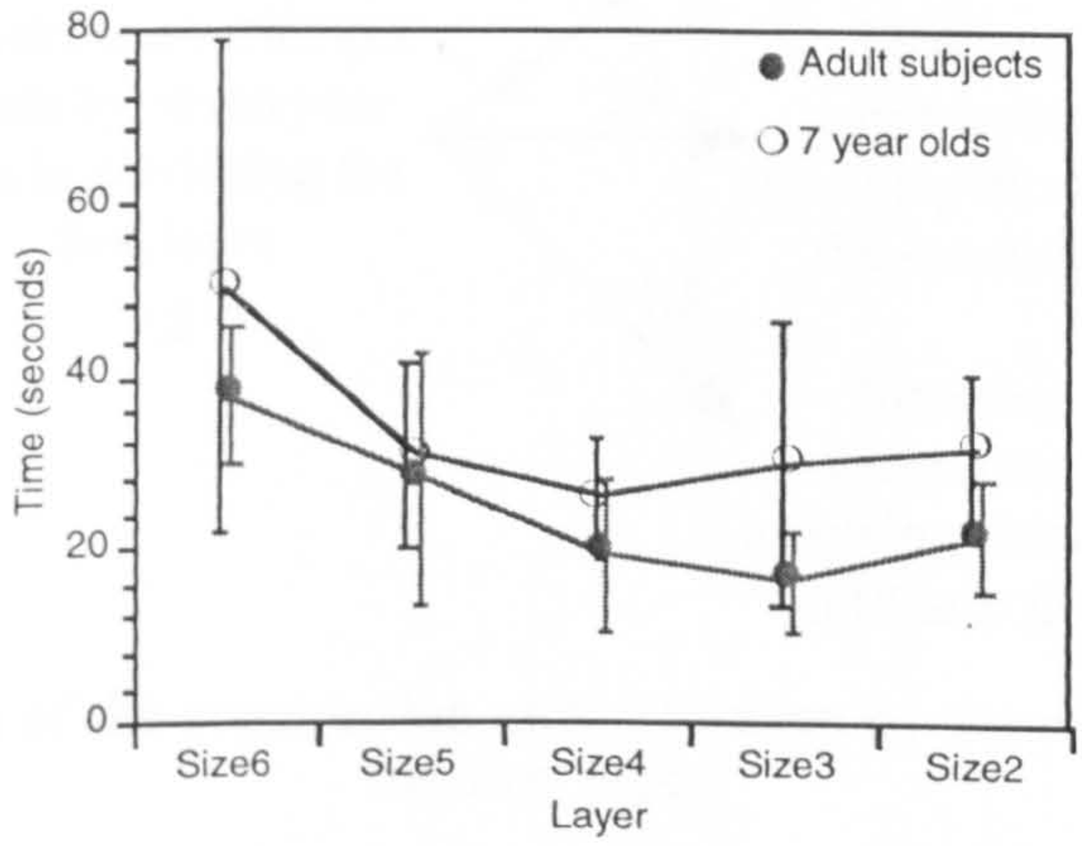


Figure 4l: Time taken to complete each of the five layers, for the adult and the seven year old subjects.

For the adult subjects, there are no reliable differences between any of the construction attempt scores and there are no reliable differences between any of the layer timing scores. For seven year old children, there is a significant difference between the number of construction attempts taken in building the first and second layer ($t(8)=2.54$, $p<0.05$).

There are no reliable differences between any other construction attempt scores, or for any layer timings.

The seven year olds produce a high number of errors when building the first layer. The errors made on the first layer account for 42% of the total task construction errors made by seven year olds. Why should this be the case? Figure 4m shows the construction attempts breakdown. The breakdown shows that an average of 1.8 of the incorrect constructions produced when constructing the first layer involve blocks of different sizes. Seven year olds average 1.8 incorrect constructions involving blocks of a different size *for all layers* (see Figure 4h). The seven year old's therefore make all their different sized incorrect constructions when constructing the first layer. This suggests that seven year old's have difficulty in selecting the largest blocks in the Tower task.

The largest two sets of blocks are the most closely related in area size. Although the size of each of the blocks differs uniformly, the size difference interacts with the actual size of the block: the two largest blocks have large areas and so they appear to be closer in size than the two smallest blocks. This is why the selection errors appear only between the two largest sets of blocks.

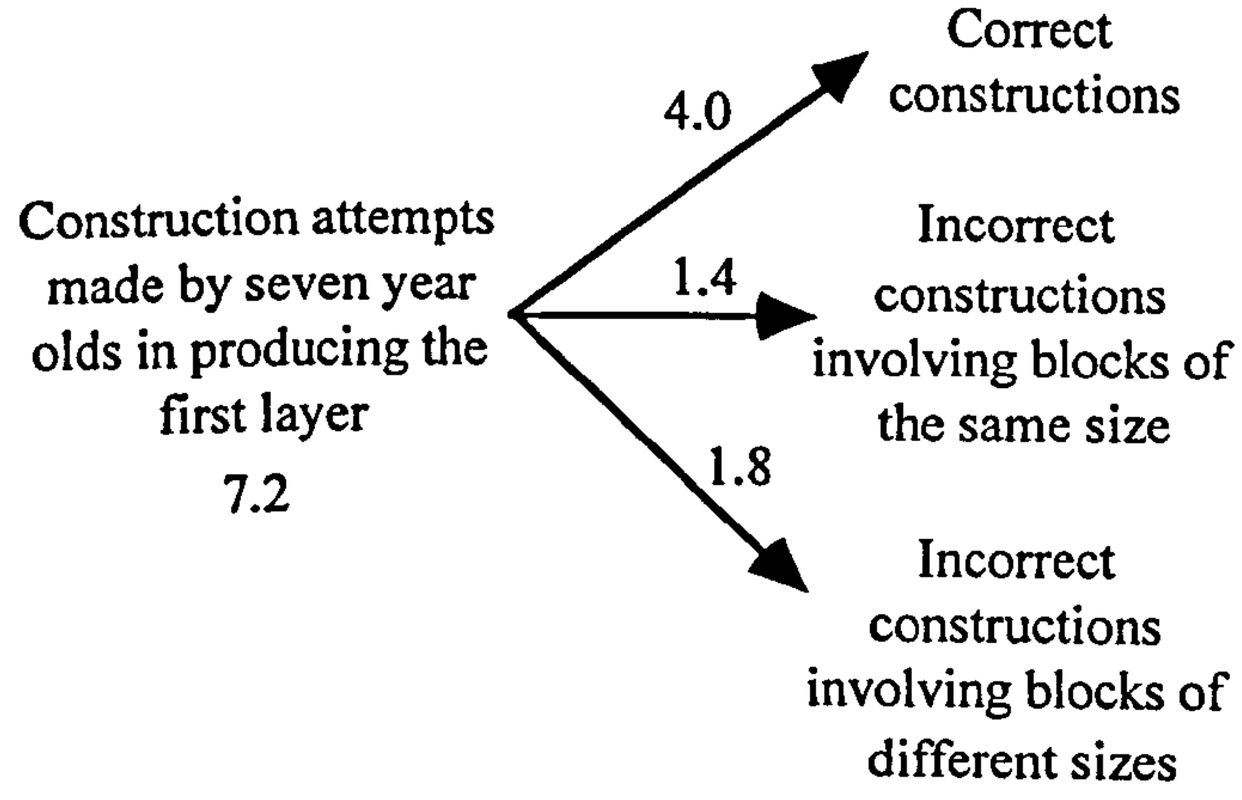


Figure 4m: Breakdown of the construction attempts taken by seven year olds in producing the first layer.

The general trend of a decrease in time taken to produce each subsequent layer may be because there is reduced visual search: as subjects produce more layers, there are less blocks to consider. The videotaped data cannot help answer this question, but modelling the task computationally should be able to indicate how much time is spent searching for blocks, for each layer.

4.4.4.4 Summary of measures

A variety of measures have been detailed in the above sections. The sections outlined nine principle measures of task behaviour (eight of these are from the overall measures section and one, the number of task appropriate features, is from the strategy measures section). These should be used to test the behaviour of any model of the task with subject behaviour. Further overall measures were outlined, as well as strategy measures and within-task measures. The most important of the additional measures are the within-task measures, which can indicate a subset of the nine measures on a layer-by-layer basis (although only time taken and construction attempts will be used throughout, because scores on the other measures per layer will be low). The time taken and construction attempts on a layer-by-layer basis should also be used to test the behaviour of a model of the task against subject behaviour.

Table 4.6 summarises the principle nine measures. The number of correct constructions is omitted because it is measured as part of the time between correct constructions. The list of measures are not restricted to those where a reliable difference exists between adults and seven year old children. The measures must provide a test of a model of task behaviour as well as to provide differences between adults and children's behaviour.

Table 4.6: Summary of the principle measures of behaviour on the Tower task. Reliable differences are indicated with a *.

	Adults	Seven year old children
Construction attempts	22.8	27.6*
Time taken	126.6 s	174.0 s*
Incorrect constructions involving blocks of the same size	1.8	5.4*
Incorrect constructions involving blocks of different sizes	0.6	1.8
Average number of times the same blocks are fit together in the same way again	0	1.0
Average number of times the same blocks are fit together in different ways	1.3	1.2
Time between correct constructions	6.0 s	8.5 s*
Time between incorrect constructions	5.0 s	2.9 s
Number of task appropriate features	1.9	1.3*

4.5 Requirements for a model of the task

At this point, it is known that the adult subjects perform better on the Tower task than seven year olds, for both overall measures and within-task measures (strategy measures are inconclusive because there is not one strategy that is conclusively better than another). The aim now is to develop a model which matches the behaviour of adults on the Tower task, and then make modifications to it to see to what extent each modification enables the model's behaviour to match that of seven year old children.

The Tower task enables multiple behavioural measures to be taken, which provide more data for the model to match subject behaviour on. This will provide a detailed analysis of model-subject comparisons, and pinpoint specific measures of behaviour which change with modifications to the model. The physical nature of the Tower task (which helps to provide the array of measures that are available) means that a significant part of the task is interaction (i.e. not cognition). The model must be able to account for the interactions with the blocks. The discussion here will relate to the need for a task simulation, and the critical behavioural aspects of the task that a model must include. Consideration of how the behaviour of the model will be matched to the behaviour of subjects is also given.

4.5.1 The need for a task simulation

Much of the behaviour in the Tower task involves low-level processing via interaction with the task environment, such as scanning the table or manipulating the blocks. Any model of the task is therefore required to include this interaction. Two methods exist for including interaction processes within a model: either include the task environment within the model, or have the model interact with an external simulation of the task environment (usually a graphical simulation written in a fairly specialised language). The development of an external task simulation has clear advantages:

1. The simulation can indicate how complex the task is and how great a role the eye and hands play, based on the number of times the model has to interact with the simulation, and for what length of time.
2. Modelling only the high-level processes involved in the task assumes that access to the external task information is problem-free. Accessing the external task information may in fact influence speed and accuracy in the task (Anderson, Matessa & Lebiere, 1997) . For example, the main source of the extra time required to complete a subitizing task (Jensen, Reese & Reese, 1950) is likely to be the extra fixations required when there is a larger number of objects.
3. It enables the parameters associated with the eye and hands to be changed easily. If the representations of the eye and hands were within the model, the parameters would be difficult to modify, because the extent to which alterations can be made is restricted by the cognitive modelling environment. For example, altering the area that the eye covers would be easy for an external simulation because the simulation should be written in a specialised language. The language of the model (e.g. rule based or connectionist) makes these changes more awkward.
4. The simulation may indicate possible aspects of the task which occur in parallel. Aligning the model/simulation behaviour against subject behaviour may reveal task processes where the model is too quick or too slow, based on matching specific behaviour to specific time points. Although a simulation is not strictly required to elicit this knowledge, in tasks which emphasise use of the eye and hands, it is likely that mismatches will be found with regard to eye/hand behaviour.
5. Modelling only the high-level processes involved in the task may mean modellers are "granting themselves unanalysed degrees of freedom in terms of choice of representation" (Anderson et al., 1997, p.442). Therefore the success of the model may simply be because of the chosen representation and not because of the high-level processes that have been modelled.

6. Including aspects of the simulation in the model may mean we take some aspects of the task for granted which are actually difficult. Many existing models may perform tasks too quickly for this reason.
7. The behaviour of the model can be viewed on the graphical representation that the simulation provides.
8. Representing low-level processes allows the examination of how to cope with physical limitations when task demands are high (Meyer & Kieras, 1997).

An external task simulation will also mean that more theoretically motivated modifications can be included. A simulation offers the opportunity to examine task behaviour that is difficult to obtain from subjects (i.e. providing further measures of behaviour). For example, the time spent performing visual search would require eye movement studies which at present are difficult to obtain because the advanced equipment required to do this is expensive. Visual search time is interesting because it can help us determine to what extent the time reduction in producing subsequent layers is due to there being less blocks on the table, and to what extent it is due to learning. A model of the task should therefore interact with an external task simulation.

4.5.2 Behavioural requirements

On first inspection, the behaviour on the Tower task seems to be a simple cycle of selecting and fitting blocks. However, complications arise when the blocks that are fit together result in a construction that does not look correct. This suggests some form of assessment of the construction that has been produced, in order to decide whether to continue or to disassemble the construction. The cycle is therefore: block selection; block fitting together; assessing the construction. These will be the central behaviours of a model of the Tower task. At each of these stages, further processes must occur (e.g. visual search when selecting blocks; aligning blocks before fitting them; disassembling a construction if it looks incorrect).

The behaviour of adults shows that there is immediate learning in the task. When adults attempt to fit two block features together, and a construction is produced, they must learn that the two features can be fit together to form a construction. However, the adults need to know what features fit together to form *correct* constructions. The adults seem to know the appearance of a correct construction. It would seem plausible that the adults, when they produce a construction they believe to be correct, reflect on it to examine what

features are attached, and what features are aligned. This knowledge can then be used when future blocks are selected to be fit together.

There is also gradual learning in the task (the reduction in time to produce subsequent layers). The importance of gradual learning in a model of the task is uncertain because the reduction in time may be due to the reduced search time as the number of blocks to select from is reduced. It is possible that the immediate learning in the task enables the subsequent layers to be constructed more quickly because new task knowledge regarding constructions is acquired.

4.5.3 Matching the model behaviour to subject behaviour

A variety of task measures have been outlined in this chapter. It will be important that a model of the task matches subject data on the task measures. The emphasis will be on the nine measures that are detailed in Table 4.6 as well as within-task measures. The strategy measures are less important. For incorrect constructions, there is a wide range of task appropriate features (i.e. features that are fit together correctly even though the construction is incorrect), and this data is noisy. A better measure is the average number of task appropriate features that are fit together correctly (this is included in the nine measures listed in Table 4.6). For correct constructions, the strategies used in building a layer are all equally effective, because each requires three correct operations. The layer strategies will still be analysed, but more emphasis will be placed on obtaining the same distribution of strategies rather than matching scores on every strategy measure.

A model of the task will need to be matched on the nine measures of behaviour listed in Table 4.6, and within-task measures. Strategy measures will be examined in terms of covering the general distribution of strategies rather than matching subjects on every measure.

4.6 Summary

Most of the important factors in producing a model of the Tower task have now been covered. These are taken as the building blocks for a model of the Tower task. The next chapter describes a model of the task, and an external simulation of the task, in detail.

5. A model and external simulation of the Tower task

In order to realise how the model and task simulation produce behaviour that is able to fit that of subjects, it is necessary to cover the details of both the model and simulation in some detail. This chapter describes both the task simulation and the model of adult behaviour on the Tower task. A description of the model and task simulation parameters that can be modified is given, as these will contribute to the modifications that will be made to the model, the task simulation, or both, in later chapters.

5.1 An overview of the model and the simulation

The model should cover all important aspects of cognition for the task, and the simulation should cover all important aspects of perceptual and motor actions for the task. For the purposes of model creation and testing, it is easier if the cognitive processes can be kept distinct from the perceptual and motor processes, because cognitive processes belong to the model and perceptual and motor processes belong to the task simulation. However, there are some areas where overlap is required (these will become apparent later, when explaining the details of the model and simulation).

The model is rule-based and contains rules for accomplishing the task as well as rules for interacting with the simulation. The simulation includes an eye and two hands which are directed by the model in order for task behaviour to occur. Information passed between the model and the simulation can be summarised as requests for action and results of action. The model will request the eye or hands to perform an action, and the results of that action are passed back to the model (e.g. for a fixation, what objects are seen; for a fit, whether the fit was completed or not). The model can then alter its behaviour based on the results of the actions, and will interact with the simulation to perform further actions.

5.2 The simulation and what it covers

The simulation covers all relevant aspects of the task, together with interfaces to pass information from the cognitive model to the external environment (provided by a simulation eye), and to manipulate the external environment (provided by simulation hands). This section covers how the simulation works and how it interacts with the cognitive model, and also what parameters can be altered within the simulation.

5.2.1 The basic simulation

The simulation is written in Garnet (Myers et al., 1990) and has both a graphical representation and an internal representation of all of the blocks, together with all of the block features. Their representation is not fully three-dimensional, although it is more complex than two-dimensional because blocks and constructions can be turned over. All objects are placed on a screen area (the "table", which is the boundary of the block area). The simulation eye is able to saccade (eye movement) and fixate (retrieve information on what is seen) on blocks and block features that are on the table, and the simulation hands are able to grab, release, rotate, fit, disassemble, and turn over blocks. Saccades take 50 ms, and fixations take 200 ms, based on a summary of the vision literature provided by Baxter and Ritter (1996). Hand movements take 550 ms based on estimates from the video analyses of the adult subjects. Behaviour of both the eye and the hands is governed by the model, which must interact with the simulation to direct the eye and hands.

The simulation eye has three regions: fovea, parafovea, and periphery. The fovea is the focal point of the eye and is very small in area (covering approximately half the area of the smallest block). The parafovea extends approximately one largest block width from the fovea boundary. The periphery covers the remainder of the table (see Figure 5.1). The areas of these regions is based on estimates from Baxter and Ritter (1996).

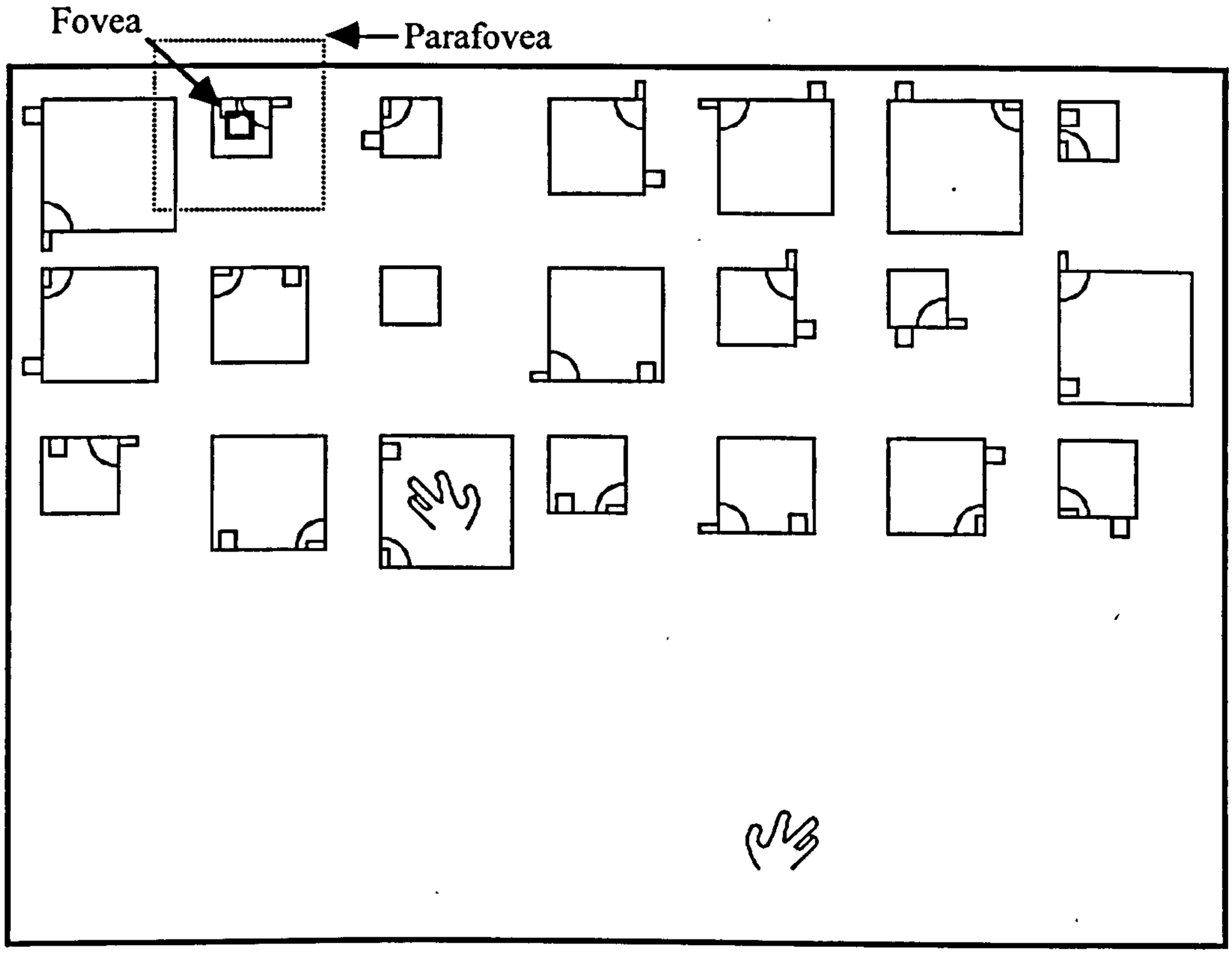


Figure 5.1: A graphical representation of the Tower task.

The block information passed from the task simulation to the model includes the size of the block, the block's location, and the features that the block has. Information regarding objects on the table is only passed for fixations; no information is recognised for saccades (this is in-line with the vision literature, e.g. Carpenter, 1977; Sekuler & Blake, 1994). The accuracy of information passed to the model depends upon the region of the simulation eye where the block lies.

The size of blocks is subject to noise for both fovea and parafovea regions. No size is given for blocks in the periphery (therefore the model only knows the location of periphery blocks). If a size was allocated to a block, it remains until either the model forces a comparison of sizes between two blocks, or the size of the block is forgotten. To simplify the simulation, location is implicit in the blocks and features. The model moves the eye and hands to blocks or features rather than to table co-ordinates.

It is possible to try to fit any block or construction feature to any other block or construction feature. If the fit can be achieved (i.e. no other block or construction impedes the fit, and no feature that is part of the blocks/constructions the model is fitting together impedes the fit),

then the model is informed that the fit was completed, and is told the identification of the new construction (so that it can now be referred to in the model). If the fit could not be achieved, then the model is informed that the fit was not completed.

When a fit is achieved, construction information is passed in the same way as block information. This means that when a new construction is produced, it has to be fixated on in order for the model to know of its size and its features. A construction feature is created when the fit means two block features become aligned to form a different feature (e.g. halfpegs aligning to form a peg). When constructions are disassembled, a fixation will remove any construction features that no longer exist.

The block and construction information alters when new constructions are produced, because the new construction may mean that some block or construction features may no longer be available for fitting (e.g. because they have been fit to another feature). The information is only updated in the model when a fixation occurs. Features can be attached (e.g. fitting a peg in a hole) or aligned (e.g. aligning quarter circles). Some can be both (quarter circles, halfpegs, and halfholes can all be both attached and aligned). The values of the attached and aligned slots are therefore updated (in the simulation, and in the model after a fixation) with the block/construction they are attached/aligned to (or reset if they are unattached/unaligned). If a block or construction feature becomes "hidden", such as placing a block or construction immediately in front of the hole feature of another block or construction, the attached and aligned values are set to "hidden" indicating that the feature is unavailable.

The hand information that is passed to the model is relatively simple. The grab and release actions pass the model notification that the action was completed. The remaining actions that the hands can perform (rotating, fitting, disassembling, turning over) rely on the relevant blocks or constructions having been grabbed (the model must ensure this is the case). When the actions are completed, notification is passed to the model. Additional information is given in two instances. First, when fitting blocks or constructions, the new construction identity is passed to the model. Second, when disassembling constructions, the disassembled construction is removed from the model. This is done to prevent attempts to access non-existent constructions if they still exist in the model as values of slots.

Discovering the result of hand actions requires a fixation. For example, when rotating a block or construction, the orientation of the block/construction features will not be updated until

the model fixates on the block/construction. No haptic information is represented in the simulation. This is a possible limitation of the model and simulation.

There is additional information that a fixation passes to the model:

1. There is a separate representation in the model for feature *types* (e.g. peg, hole). Every block feature that is represented in the model will be an instance of a feature type (e.g. peg123 is an instance of feature type peg and belongs to block ABC). All features that are passed to the model will have their associated feature type noted as being task relevant. If the model no longer has knowledge of block features of a specific type, the feature can still be searched for, because the model knows that this type of feature is relevant to the task.
2. All of the different block sizes that are seen are ordered. This means the model has knowledge of how many different size blocks there are, and in what order the layers get stacked.

It is worth noting that whilst Garnet is a specialised language for this kind of simulation, it is still fairly rigid because it does not contain built-in functions that are able to fully manipulate the graphical objects that we define. For example, when fitting a feature of one block to the feature of another block, Garnet cannot identify obstructing objects. It supplies general functions for manipulating objects which have to be used extensively to build self-made functions for this kind of check. Similarly, functions have to be built to rotate blocks/constructions, turn blocks/constructions upside down, move blocks/constructions, and so on.

5.2.2 Simulation parameters

The objects within the simulation (blocks, features) do not change in their characteristics across ages, in line with the experiments across ages which all use the same set of blocks. However, there are several parameters within the simulation that can be altered in order to examine their effect upon the behaviour of the model:

1. Size of the fovea.
2. Size of the parafovea.
3. Probability of inaccuracy when fitting blocks together internally (i.e. imaging how the blocks fit together without physically attempting fitting the blocks).
4. Probability of inaccuracy in internally determining whether anything obstructs a fit between two blocks (i.e. imaging whether or not an intended fit is obstructed without physically attempting to fit the blocks).

5.2.3 Comparison of the simulation with other visual interfaces

The ACT-R architecture has its own visual interface. When this was first presented (Matessa & Anderson, 1996), the interface was limited in its set of operations and in the set of domains it could be applied to (Byrne & Anderson, 1997). These are the central reasons why the interface was not used for the Tower task. The interface has since been improved, and is integrated fully into the ACT-R architecture as ACT-R/PM (Anderson, Matessa & Lebiere, 1997). ACT-R/PM is able to cater for both perceptual and motor actions. The comparisons with other interfaces will concentrate on the eye, since the hands in the Tower simulation are elementary (e.g. they do not account for haptic information).

The ACT-R/PM visual interface is similar to the Tower visual interface because its simulation environment is separate from the model, with interaction between the two involving requests for action and results of actions (generally involving the creation of Declarative Memory Elements [DME's]). However, the eye in ACT-R/PM is not split into regions but has a variable "spotlight of attention", and a variable "scale of features". This means the visual field is variable and so is the level of detail that vision sees (e.g. seeing lines and angles versus seeing a square). In ACT-R/PM the area of the visual field is normally the full screen. The Tower simulation concentrates more on the biological aspects of the eye and hence has three regions: because of the size of the fovea and parafovea, it would be difficult for everything on a computer screen to be seen in detail. This may be offset by the scaling of features that ACT-R/PM has. Scaling of features is unnecessary in the Tower task, at least for adults and children of seven years old. ACT-R/PM does not differentiate saccades and fixations, and so the assumption is that a fixation occurs after every saccade. The timing for the full action (i.e. both a saccade *and* a fixation) is 185 ms which is slightly below the ≈250 ms used by the Tower model and simulation.

The EPIC architecture includes both an eye and hands, as an architecture of low-level processing. Their simulation eye is very similar to that in the Tower simulation. It has the same three regions with the same associated timing of ≈250 ms (Kieras & Meyer, 1996), although this depends on the visual properties of the objects seen. The 250 ms timing constitutes the detection of shape information (≈100 ms) and then the encoding of additional perceptual properties into visual working memory (≈150 ms). The timing for the Tower simulations' eye is to saccade and then fixate – the fixation creates the objects seen and places them directly into the models' working memory through the creation of DME's. The level of information that the EPIC simulation eye passes is unclear, although as it is an architecture for low-level processing, it is assumed that the information is at the most sensible lowest

level (e.g. seeing lines and angles rather than a square). For EPIC, ACT-R/PM, and the Tower simulation, location information is directly available once an object is seen.

The Tower simulation bears similarities to both ACT-R/PM and EPIC, and includes properties of both. One limitation that may require addressing in the future is the assumption that features in the Tower task that are seen by the simulation eye require no further processing. This implies that objects such as pegs are not seen as their component parts but as a whole object which can be placed directly into the working memory of the model. This appears not to be the case with EPIC, which is presumed to encode low-level visual properties; in ACT-R/PM the level of detail is governed by the scale of features parameter. This means that ACT-R/PM has the option of an object either being viewed as itself or as its component parts. This limitation is not addressed within this thesis.

5.3 The creation of the model

Most computational models have to include several aspects of psychology because the tasks that they model require subjects to use various psychological mechanisms when completing the task. For example, models of word recognition, a task where a string of letters is presented and subjects affirm whether they believe the string to be a word (e.g. Andrews, 1989; Grainger & Segui, 1990), must use at least two types of memory (one for the presented string and the other for some database of known words) and some decision process. This means that any prospective modeller must be aware of these mechanisms and how to implement them in their chosen modelling language. However, there are general modelling architectures that have been developed ("cognitive architectures"), in which some of these mechanisms are already implemented (the implementation is not always the same across cognitive architectures). Issues regarding the implementation of some of the various psychological mechanisms that may be pertinent to the task can therefore be avoided by choosing to develop a model within one of the available cognitive architectures.

5.3.1 Summary of cognitive architectures

There are two broad categories of cognitive architecture: symbolic and sub-symbolic. There is much debate between proponents of each architecture regarding which aspects of human learning and behaviour that each is able to model (e.g. Bechtel, 1993; Fodor & Pylyshyn, 1988), and there are also summaries of the architectures and the debates (e.g. McLaughlin, 1993). Rather than reproducing these arguments, reasons for selecting an architecture which has symbolic properties are given.

The main difference between symbolic and sub-symbolic architectures, with regard to how the Tower task will be modelled, is implicit in the name: sub-symbolic architectures do not attach familiar symbols to their representations. This presents two immediate difficulties: first, it is not obvious what has been learnt when the model performs the task; second, it is difficult to examine the reasoning behind choices the model makes when performing the task (a symbolic architecture will be able to express both of these using familiar tokens). This information is required because it enables comparisons between different Tower models to be made. Representing the task world symbolically also means that input to and output from the model is in a meaningful representation which can be easily interpreted by both the modeller and the reader.

Rule-based models should be particularly suited to knowledge modifications because of their modularity (Klahr, 1984a). Production rules can be inserted, removed, or changed without having to take into account all of the other rules in the model.

Given that there is a requirement for some symbolic element to the computational model, three cognitive architectures stand out: Soar (Laird, Newell & Rosenbloom, 1987), ACT-R (Anderson, 1993), and EPIC (Meyer & Kieras, 1997). All three are rule-based, but only Soar and ACT-R include learning – EPIC models low-level tasks and does not yet include any learning mechanisms. The Tower task shows within-task learning, and so the EPIC architecture must be ruled out, leaving Soar and ACT-R. There are various comparisons between Soar and ACT-R available (e.g. Jones, 1996; Rieman, Lewis, Young & Polson, 1994). A brief comparison will be given here.

Soar and ACT-R can be seen as similar in several ways. Both reduce much of human behaviour to problem solving. Soar does this explicitly, being based upon Newell's information processing theory of problem solving (e.g. Newell, 1968), whereas ACT-R merely implies problem solving behaviour by being goal directed. Both are production systems which have two kinds of memory, declarative (facts) and procedural (rules). In both architectures these memories are conceptually infinite, with no provision being made for the removal of any memory item in ACT-R (the Soar architecture does perform removal of declarative memory, which therefore can be seen as short-term). Manipulation of declarative memory can be accomplished by adding new items or changing existing ones. For procedural memory, rules may only be added.

The course of processing involves moving from an initial state to a specified goal state. Both ACT-R and Soar maintain a goal stack, where the goal at the top of the stack is the current goal of the system. Movement between the initial and goal states usually involves the creation of sub-goals in order to accomplish the various parts leading up to the satisfaction of the goal. Both ACT-R and Soar achieve this through the goal stack, where each subsequent sub-goal is placed at the top of the stack and hence becomes the focus of the system. The goals must be satisfied in a serial manner, and in the reverse of the order they appear in the stack. However, both ACT-R and Soar are capable of removing intermediate sub-goals should the current goal resolve a goal that is lower down in the stack.

There are also fundamental differences between the two architectures. ACT-R is a hybrid architecture because its declarative and procedural knowledge have both symbolic and sub-symbolic aspects to them (Lebiere, Wallach & Taatgen, 1998). Movement between states is done in ACT-R by firing productions, which *may* change the state. Where the conditions of several productions are met, a conflict resolution mechanism selects the production which ACT-R estimates to have the highest gain.

Soar is a solely symbolic architecture. Movement between states is achieved by applying an operator to the state. All productions whose conditions are met will fire, and propose operators or add information to declarative memory (which in turn may cause more productions to fire). When no more productions can fire, an operator is selected. This whole process is called a decision cycle. Where an operator cannot be selected (e.g. due to several operators conflicting each other), a sub-goal is created with a goal to decide upon one single operator.

Behaviour on the Tower task can be modelled in both architectures. The central behaviour is deciding how to fit blocks together. Each method of fitting blocks could be represented as a production in ACT-R or an operator in Soar. However, the behaviour on the Tower also includes learning. Indeed, modelling cognitive development per se places emphasis on the learning mechanisms of the architectures, because learning occurs during development. It is therefore important to consider the learning mechanisms that each architecture uses.

5.3.1.1 Learning in ACT-R

ACT-R learning involves both declarative and procedural memory. Much of this learning involves activation values, which can affect the time taken for productions to fire. Each item in declarative memory has an activation value that changes based upon how often it has been

used, and how strongly it is associated with other items that are being used. This means that the more often an item in declarative memory is used, the higher its activation will become. The more strongly associated an item is with ones that are being used, the more chance that item has for having its activation raised.

Productions have an associated strength which alters based upon how often the production has been used. The strength of productions influences the activation of all the declarative memory items that are in the condition of the production.

Each production also has an expected gain value (which is used for conflict resolution; the production with the highest expected gain is selected when several productions are instantiated). The more often the production meets with later success (e.g. the sub-goal is completed successfully), the higher this value may become. Similarly it can decrease if the production meets with failure. New productions can be created (by using an analogy mechanism) in the action component of a production.

5.3.1.2 Learning in Soar

Learning in Soar occurs only for production memory. New rules are created by the architecture whenever a sub-goal is resolved, such that when next encountering the same situation, the new production fires without the need to enter a new sub-goal. This can lead to long-term declarative memory learning, for long-term declarative information is represented solely as the result of procedural memory.

5.3.1.3 Learning in the Tower task

The within-task learning that occurs in the Tower task is presumed to chiefly involve learning how blocks fit together (testing this hypothesis is one further reason for modelling the task). ACT-R offers several methods by which this learning could be achieved: learning more specific productions (e.g. fitting a peg in a hole *and* aligning the quarter circles, rather than just fitting a peg in a hole); adjusting the strength of productions that result in success; strengthening the activation between block features that have produced a successful construction in the past. The Soar architecture can only learn new rules and therefore models the behaviour by learning more specific productions (although other learning methods have been implemented within the Soar architecture).

It now seems apparent that within the ACT-R architecture there are more methods for modelling learning in the Tower task than there are within Soar. Therefore there will be more modifications to explore by adjusting the parameters within the learning mechanisms to see what effect they have on the behaviour of the model. ACT-R is also written in the same language as the simulation (Garnet and ACT-R are written in Macintosh Common Lisp), so interactions between the model and simulation are much easier to implement, and modifications to simulate development are easier to implement.

5.3.2 An overview of ACT-R

The two types of knowledge in ACT-R are procedural and declarative. Declarative knowledge in ACT-R consists of declarations of objects types, and then the creation of specific instances of those object types. The object type can be viewed in a similar manner to frames (Minsky, 1975). An example object type is:

DMEType geometric-shape comprises fits-in

which declares geometric-shape as an object type having the two slots *comprises* and *fits-in*. A specific instance of an object type is called a Declarative Memory Element (DME). For example, a DME named *peg* can be created as an instance of the geometric-shape object type:

peg ISA geometric-shape comprises halfpeg fits-in hole

The *peg* DME has the values of the *comprises* and *fits-in* slots being *halfpeg* and *hole* respectively. If the value of a slot is not known when a DME is created, the slot can be omitted (its value will default to nil, a null-value). When a fixation occurs, a DME is created for every relevant block and block feature that is seen.

Procedural knowledge is defined as production rules, the conditions of which examine declarative memory, and the actions of which usually change declarative memory in some way. Goals can be pushed onto the goal stack, meaning they become the active goal, or popped off the stack, meaning the next goal down in the stack becomes the active goal (if none exist, behaviour terminates). Each production rule *must* have as its first condition a match to a goal. This can restrict the applicability of production rules such that they only apply to specific goals. Variables in productions are specified by being preceded with an "=". The variable name "goal" is reserved and must always be used when assigning the current goal in the first condition of a production rule. A production rule called *search-for-pegs-and-holes* could look like:

```
search-for-pegs-and-holes
        =goal>
                ISA             decide-search
```

```
        =feature1>
                ISA             geometric-shape
                fits-in         =feature2
        =feature2>
                ISA             geometric-shape
                fits-in         =feature1
    ==>
        =newgoal>
                ISA             look-for-features
                feature1        =feature1
                feature2        =feature2
    !push!                      =newgoal
```

The production rule has three conditions which must be matched (i.e. for each condition, there must be a corresponding DME in declarative memory). First, the current goal DME must be of the object type *decide-search*. Second, there must be a *geometric-shape* DME which has a value assigned to its *fits-in* slot (DME's with null-values in this slot will not get matched). Third, there must be a *geometric-shape* DME whose *name* is the same as the value in the *fits-in* slot of the DME in the second condition. The *fits-in* slot of this DME must be the same as the *name* of the DME that was assigned in the second condition.

The production rule has one action: the creation of a new DME of type *look-for-features*, which has two of its slots assigned values. The DME is pushed onto the goal stack meaning it will become the active goal should this production rule fire.

A production rule can fire if all of its conditions are matched in declarative memory. If this is the case, the production rule is *instantiated*. More than one production rule may match the current knowledge and the active goal. In addition, the *same* production rule may be matched by different DME's. All production rules that can fire are placed in a conflict set. ACT-R will only fire one of the production rules that are in the conflict set (the procedure of creating the conflict set, selecting a production rule and firing it is called a *cycle*). Before detailing how a single production rule is selected, DME's must be covered in a little more detail.

Each DME has an activation value. When a DME is first created, its activation is set by the *base level activation* parameter. A *retrieval threshold* parameter can be set to determine at what activation a DME must be in order to be considered for matching in the conditions of

production rules (the one exception is the goal DME which is always matched). Only DME's whose activation is greater than the retrieval threshold can be matched in the conditions of production rules (all DME's above retrieval threshold are said to be in *short-term memory*, or STM). This is consistent with the view that "we can think of [declarative] memory as a vast body of knowledge, only a small part of which is active at any moment. The rest is passive. Short-term memory corresponds to the active part, long-term memory to the passive part." (Atkinson, Atkinson, Smith, Bem & Hilgard, 1990, p282).

For every cycle, if a DME has slot values that are the same as any slot values in the goal, then its activation will be increased. The size of the increase is based on the *goal activation* parameter, the number of slot values in the goal, and the *strength of association* between the DME and the goal DME. The goal activation is divided amongst the number of slot values in the goal, and is multiplied by the strength of association. This follows the equation

$$A_i = B + \Sigma(W_j * S_{ji})$$

where A_i represents the new activation of the DME that shares at least one slot value with the goal DME, B represents the base level activation, W_j represents the amount of activation passed from the goal (the goal activation divided by the number of slot values in the goal DME), and S_{ji} represents the *strength of association* between the two DME's. It follows that if a DME shares two values with the goal it receives double the increase in activation. The strength of association is initially set as:

log(number of DME's in declarative memory)
– log(number of DME's containing the shared value).

Every production rule has an expected gain value associated with it, which reflects how well the production rule can contribute to achieving the main goal. The value is calculated by PG – C, where P is the probability that the production rule will help in achieving the goal, G is a *goal* parameter, and C is the expected cost of firing the production rule. P is calculated by multiplying the probability that the production achieves its intended effect with the probability that the subsequent productions achieve their intended effect. C is calculated by adding the cost of firing the production rule with the cost of firing the subsequent production rules.

Maintaining statistics for the P and C values requires definitions of which productions denote a success and which productions denote a failure. This can be done by placing a success or a failure flag on a production. ACT-R then updates either the success statistics or the failure statistics for all productions that fired from the previous success/failure to the present

success/failure. The expected gain values for the productions then change accordingly, based on whether the production was successful or not, and the cost of success/failure, based on the time taken. This means in theory a production could be successful yet have its expected gain value reduced, because the success took so long to achieve.

When several production rules are in the conflict set, the one with the highest expected gain value is selected. However, if the same production is instantiated several times, the expected gain value for each instantiation will be the same. In this case, ACT-R selects the instantiation which has the most active DME's in its condition.

Parameters exist in ACT-R whereby expected gain values for productions can be subjected to noise. This enables their values to fluctuate up or down by a random amount so that behaviour, in terms of production selection, may be variable.

Timing estimates for productions are calculated from the sum of the times to match all DME's in the condition of a production rule, plus a default production timing parameter, *a*. ACT-R therefore produces absolute timing predictions for tasks. Individual production rules can have timing estimates that differ from the default. This means that interactions with the task simulation (e.g. to pick up a block) can take longer than the default production rule timing estimate.

5.4 The ACT-R model of adult behaviour

The model is written in version 3.0 of ACT-R. It will be described in three parts. First, the task representation is briefly described, together with how the model obtains information from the task simulation. Second, the ACT-R mechanisms that are used by the model will be described, together with any mechanisms used which are outside the ACT-R architecture. Third, the production rules that describe the behaviour of the task are detailed. Once these three have been detailed, a summary is given which tells the reader what learning occurs in the model, and which production rule module the learning occurs in.

5.4.1 Representation and acquisition of information in the model

The simulation representations are high-level, and therefore the model has the same representation for blocks, constructions, and features as the simulation has, although the model has an additional feature type representation for every feature in the task. This means there is no process which transforms low-level visual stimuli into high-level representations. The assumption that there is familiarity with the shapes of the blocks, constructions, and

features seems to be true of adults and seven year olds, based on the video analyses of their behaviour.

When the model requests the eye to saccade, the blocks or constructions that are on the table may now be in a different region of the eye. As stated previously, the size of blocks and constructions retain their original values, unless they were not originally set a size (because they were initially in the peripheral region, in which case only a location is given). However, the model has a *decay* mechanism such that values are reset when the model representations of the objects decay so much that they fall below the ACT-R retrieval threshold. Decay will be covered in the next section.

5.4.2 Architectural mechanisms used by the model

The model incorporates three learning mechanisms. These will be covered in detail later:

1. New productions are learnt through both proceduralisation of declarative knowledge, and through reflecting upon task behaviour.
2. Factual knowledge about the task is learnt through the creation of DME's.
3. Sub-symbolic learning occurs for the expected gain values of production rules, so that the model learns which production rules to use in which situations, based on the success and failure rates of the contending productions.

Learning productions in ACT-R involves use of the analogy mechanism. This requires an example goal and an example solution to be created in declarative memory. These are difficult to describe for this task. However, the main problem with the mechanism is that the analogised production rule is based on the *values* of DME slots rather than which slots to pay attention to. When a goal DME has many slots, only some of which are important at this time, it cannot be specified which slots of the goal should be considered in creating the new production rule[2] . Only the values (of slots) can be specified. This means the new production may be forced to contain slots which may hinder progress (because they share the same values as other slots).

The problems with the analogy mechanism meant a separate production rule learning mechanism was developed outside of the ACT-R architecture. The mechanism enables descriptions of the important slots of the goal to be specified, rather than the important slot values. The learning mechanism is based on the analogy mechanism, but is versatile enough to

---

2 This facility is available in the most recent version of ACT-R (Anderson & Lebiere, 1998).

cope with both types of rule learning discussed in item 1 above. The learning mechanism creates new production rules immediately (as in Soar), rather than analogising from example and solution DME's (as in the current ACT-R learning mechanism).

Activation of elements is raised by two mechanisms:

1. On every *fixation*, the activation of all objects passed to the model is raised to the base level activation (i.e. all constructions, but only those features that are in the fovea and parafovea). If a DME is already above base level, then no change is made to its activation. This mechanism is outside of the ACT-R architecture, because ACT-R 3.0 does not have an advanced simulation environment.[3] Raising activation for all objects seen bears similarities with connectionist models of vision where activation is passed into the network whenever features are present (e.g. McClelland & Rumelhart, 1981).
2. On every *cycle*, any DME's which share values with those of goal slots have their activation raised. This is based on the equation

$$A_i = A_i + \Sigma(W_j * S_{ji})$$

which is identical to the standard ACT-R equation except the increment is added to the current activation of the DME rather than the base level activation. This is because the model includes decay: using the standard mechanism would mean activation always remained at base level or more.

Activation of elements is lowered on every *cycle*. A decay mechanism (which is not part of the ACT-R architecture) acts upon all constructions and construction features, and also all of the completed goals in the model. All other items in declarative memory do not get subjected to decay. This is because they are either task relevant knowledge (such as knowing that pegs fit into holes) which should always remain active, or they are DME's which will never be used again (such as sub-goals for rotating blocks). If task relevant visual stimuli, such as pegs and holes, cause the activation of their associated DME's to rise every time they are fixated upon, then all task relevant facts will remain fairly active. This is compatible with other activation based models such as the Interactive Activation Model (McClelland & Rumelhart, 1981), which passes activation to letters which possess components that are consistent with the visual stimuli. For example, a vertical bar at the right of the letter would pass activation to letters H, M, and N, all of whose rightmost components are vertical bars.

---

3 The most recent version of ACT-R, which includes ACT-R/PM, raises activation of fixated objects by treating the object in focus as an activation source (Byrne, personal communication, 1998).

The activation of DME's that are subject to decay is lowered by an amount determined by the decay parameter *d*. The decay function is logarithmic following the equation

$$A_i = A_i - \log(A_i)^{-d}.$$

The raising and lowering of activation occurs simultaneously. A cycle raises the activation of blocks, constructions, and features that are part of the goal, and then subjects them to decay.

5.4.3 The production rules that cause the model's behaviour

The ACT-R model has 317 rules. A simplified structure of the behaviour of the ACT-R cognitive model is shown in Figure 5.2. Each rectangle represents a behaviour module which is a suite of production rules accomplishing the behaviour described by the text in the rectangle. The directional arrows show the course of processing between modules.

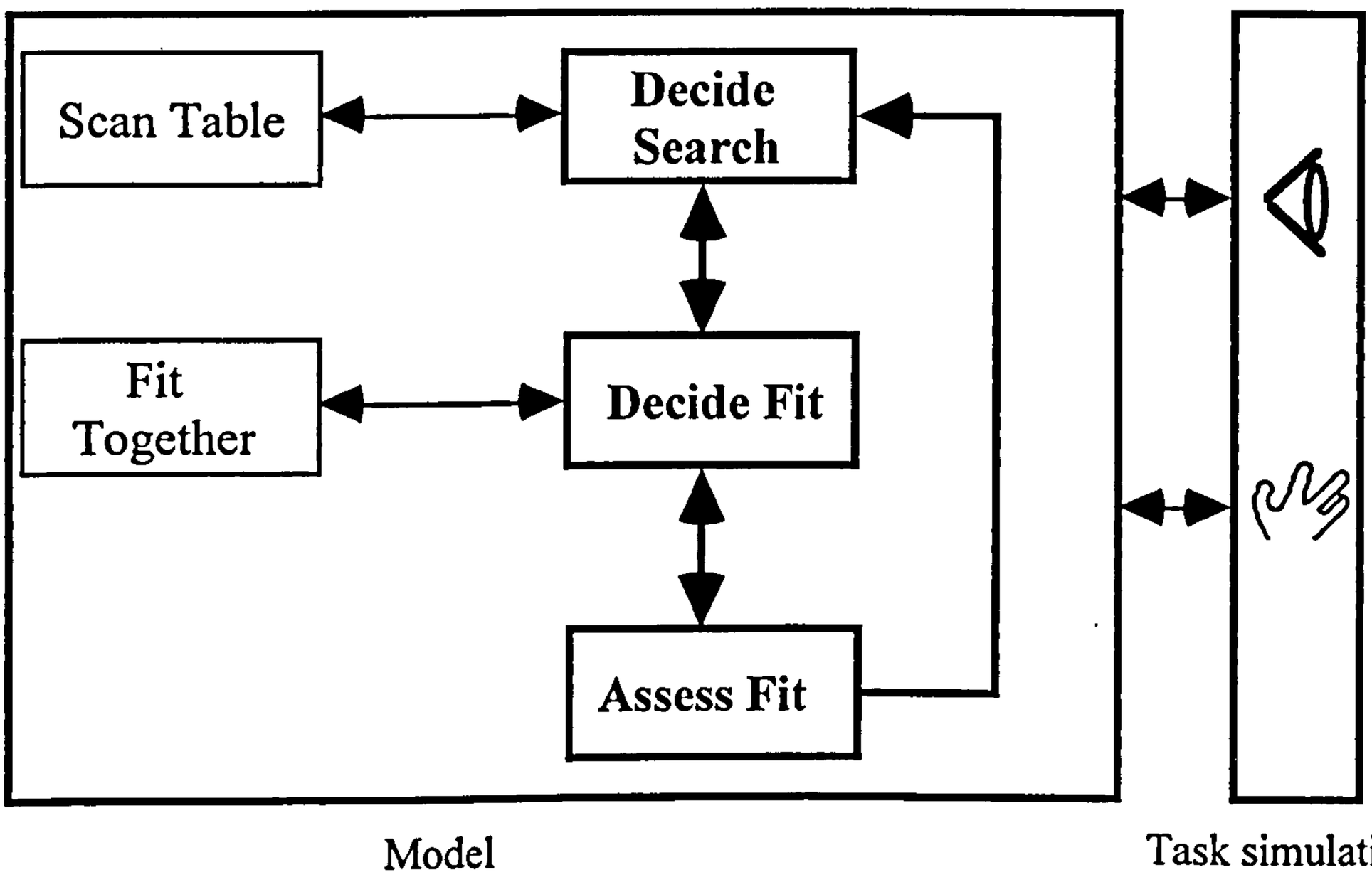


Figure 5.2: The major production rule modules in the ACT-R cognitive model of the Tower task (central modules are in bold).

Each module of production rules all refer to a goal that is specific to that module. The modules in bold (decide search, decide fit, assess fit) are the central behaviour modules: their behaviours are closely linked. For ease of modelling, the *same* goal is used for each of the three central modules; a goal-type slot ensures that only productions that refer to the specific goal-type (decide search, decide fit or assess fit) are able to be matched. For example, when deciding how to fit two blocks or constructions together, the goal-type slot is set to Decide Fit which means only those productions in the Decide Fit module are able to be matched.

The goal for the three central modules consists of slots for the blocks or constructions in the left and right hands, together with slots for left hand size and features, and right hand size and features. These indicate size and features to search for when no block or construction has been picked up, or the features to fit by when the model is deciding how to fit the selected blocks or constructions.

The other two modules (scan table, fit together) can be set as sub-goals. Upon completion of the sub-goal, control is passed back to the previous goal (because it will now be on top of the goal stack). For example, after it has been decided how to fit the blocks that are held, they are fit together by creating a Fit Together sub-goal, which carries out the fit. The sub-goal is taken off the goal stack when it is completed, meaning that the active goal returns to being Decide Fit.

The modules that interact with the simulation can also be set as sub-goals from any behaviour module. For example, when in the Fit Together behaviour module, the model may need to grab the blocks if they are not already held. Productions exist in the Fit Together module which set a new sub-goal to Grab Block. Once the block has been picked up, the sub-goal is completed and control is passed back to the Fit Together module. Although the interaction modules are not represented in Figure 5.2 (there are too many of them), their communication with the simulation eye and hands is represented with bi-directional links between the model boundary and the simulation boundary.

To understand how task behaviour develops, the behaviour modules must be described in a little more detail. There are various other behaviour modules for which no description is given, because they are not central to the understanding of the models' behaviour. The descriptions of the five behaviour modules shown in Figure 5.2 will either implicitly or explicitly name these other behaviour modules.

5.4.3.1 Scan Table

When the model is initiated, there are several productions that decide what sub-goals to set. The first sub-goal is to scan the table so that a basic representation of the objects on the table is created in the model. Each scan involves a saccade and a fixation. The model saccades to blocks or constructions which have not been fixated upon before. If there are several candidates, one is chosen at random (though objects in the periphery are preferred). The blocks and constructions which have already been fixated upon are known to the model because a goal to fixate on them will have been carried out. The completed fixation goals are subject to decay so the model eventually forgets about fixating on some objects.

When the Scan Table sub-goal is completed, the previous goal becomes active again. When beginning the task, there are three options for what to do next: If the model knows that blocks of the same size fit together (the model begins with this knowledge, because the adult subjects do) then a sub-goal is set to gather blocks of the largest size, before setting the goal to Decide Search. If the model has no knowledge that blocks of the same size fit together, then it either analyses the objects seen (this can result in new knowledge being added; for example, seeing lots of pegs and holes, and having knowledge that pegs can fit in holes, the model may learn declarative knowledge that pegs fitting in holes is a task appropriate behaviour), or sets the goal to Decide Search even though little task knowledge is known.

5.4.3.2 Decide Search

Decisions about the criteria to use for selecting blocks needs to combine the declarative knowledge about task specific information (e.g. what features can be attached, what features can be aligned), and the declarative knowledge about what objects have been seen on the table. Blocks can be selected by size, by one feature, or both. Each requires declarative knowledge regarding the task and the objects seen. To select by size, the model must know that blocks of the same size fit together, and blocks of the size in question must have been seen. Similarly, to select by features, the model must know that the features can either attach to each other (e.g. pegs and holes), or align to each other (e.g. quarter circles), and the model must have seen the features in question.

The model begins with the following declarative knowledge: blocks of the same size fit together; pegs and hole attach; halfpegs and halfholes attach; quarter circle indents and depressions attach; halfpegs align; halfholes align; quarter circles align; semi-circles align to make a circle. This initial knowledge is based on the adult behaviour where only behaviour conforming to this knowledge is seen.

The initial knowledge together with the productions in Decide Search allow the model to search for blocks or constructions of a specific size, having one specific feature, or both. Two or more features on a block cannot be searched for simultaneously. Wolfe (1994) suggests that although multiple feature search may occur, it may also be noisy.

Once the search criteria has been decided, a sub-goal is set to look for constructions matching that criteria. The sub-goal terminates when it has found blocks or constructions that match the search criteria, or when it cannot find any after scanning the table. Therefore when

returning to the Decide Search goal, the model either proceeds to the Decide Fit goal, or alters the search criteria.

5.4.3.3 Decide Fit

Deciding how to fit blocks does not use the criteria on which the blocks were selected. The decision is based on what features the blocks have and what task knowledge is known about the particular features. The Decide Fit goal only considers fitting by one feature (this is because co-ordination of features would result in very few errors, yet adult subjects produce 2.8 errors on average). However, productions can be created which co-ordinate features (this requires reflection – see Assess Fit).

There are five standard productions which decide what features to fit together. Three of these use the same featural task knowledge as Decide Search. Production one states that if each block has a feature, and knowledge exists stating that the feature of one block can be attached to the feature of the other, then attach the features. Production two states that if each block has a feature, and knowledge exists stating that the feature of one block can be aligned to the feature of the other, then align the features. Production three is the same as production two but is for quarter circles (because they can align in two different ways whereas halfpegs and halfholes cannot).

Productions four and five do not use any task knowledge: production four will select any two of the block features and attach them; production five will select any two of the block features and align them. These productions have a lower expected gain value than productions one to three, because they only want to be used as a last resort. For the current task knowledge that the model begins with, productions four and five should never be used. For any two blocks, there will always be a combination of features for which task knowledge about fitting is known. The productions will only be used if all combinations of block features for which there is task knowledge have been exhausted.

None of the productions are feature specific (i.e. they will match for any feature fitting the criteria), except for production three.

All five standard productions also require further task knowledge: First, the model must know at what orientation the features have to be. Knowledge exists regarding orientations for every set of features that the model has initial task knowledge of. A combination of features is selected based on this task knowledge. If no orientation knowledge is known for the combination of features, a fit attempt will be made at the current orientations of the features.

Second, the features selected must not have been used for a previous fit. When a Fit Together sub-goal for specific blocks and features is terminated, the sub-goal DME remains in memory but is subject to decay. If there is a fit together DME matching the same blocks and features, and it is above retrieval threshold, then the particular combination of block features will not be considered for fitting.

Although decisions on fitting blocks do not take into account the search criteria, if the same production is instantiated several times, the most active one will be selected. Features that were part of the search criteria should be more active than other block features, because they were part of the Decide Search goal. When a decision on which features to fit has been made, the Fit Together sub-goal is created, stating what features to fit and at what orientation.

The Fit Together goal will either return a successful fit (this means a fit occurred; it does not mean the new construction is correct) or an unsuccessful fit. When a fit is successful the goal is changed to Assess Fit. When a fit is unsuccessful, the attempt is flagged as a failure, so that the expected gain values of productions leading up to the failure will be reduced. Following an unsuccessful fit, either two alternatives features will be selected to be fit (if the model does not know the existence of a Fit Together sub-goal involving the block features), or the blocks are discarded and the goal is set to Decide Search.

5.4.3.4 Fit Together

Before being able to try to fit the features, the Fit Together goal must ensure that the blocks or constructions are held in the hands, and that the features are rotated to the correct orientations. Adults always produce constructions that are flush on their outer edges, so the Fit Together goal performs an internal check on the intended fit. The internal fit perceives whether or not the resulting construction will have flush outer edges without actually carrying out the fitting of the blocks. The internal fit procedure corresponds to a mental process that subjects carry out (which is why they never produce constructions which are not flush on their outer edges).

If the internal fit perceives the intended construction to have flush outer edges, then two further productions compete with each other to decide whether or not to check whether the fit will be obstructed by any objects. Checking if an object obstructs the fit is similar to the internal fit process: the fit is carried out internally rather than physically. Should the potential construction be considered not to have flush outer edges, or there is considered to be an object obstructing the fit, then the sub-goal is terminated without physically attempting to

fit the blocks; the sub-goal returns that no fit occurred. This will lead Decide Fit to flag the fit as unsuccessful.

If all the checks are successful, the blocks are fit together. If the fit resulted in a construction, then the declarative knowledge that was used to decide the features to fit by is proceduralised: a production is created in Decide Fit specifying that if in the goal there are two blocks, and they have the specified pair of features as part of their feature set, then create a sub-goal to fit the blocks using the specified pair of features. This production will be similar to the one that fired in Decide Fit to select the features to fit by, but it is for a *specific* paired combination of features.

A new production is created even if the construction is later determined to be incorrect. It is through the expected gain values of productions that the model will learn to use the productions which result in a correct construction. The sub-goal is then terminated successfully.

If the fit attempt failed (e.g. there was an obstructing object which was not identified because no obstruction check was performed) then the sub-goal is terminated unsuccessfully.

5.4.3.5 Assess Fit

The Assess Fit goal determines whether a construction that has just been created is perceived as being correct or incorrect. A construction is perceived as being correct if it is both flush on its outer edges and has its quarter circles are aligned. This is consistent with adults, who always reject constructions which do not fulfil these criteria.

When a construction is perceived as being correct, a Possible Reflect sub-goal is set to decide whether to reflect on the appearance of the new construction. Reflection, if done, requires realising that the pair of features that were fit together also led to another pair of features either becoming aligned or attached. Task knowledge must exist that states the extra pair of features can be attached or aligned (the same knowledge is used by productions in Decide Fit and Decide Search). If this knowledge is not present, no reflection can take place.

If reflection is possible, a new production is created that co-ordinates the two pairs of fitting features. This is placed in the Decide Fit module. The production states that if the hands are holding blocks that have the two sets of paired features, then create a Fit Together sub-goal that takes into account both pairs of features when fitting. The production is therefore for

two *specific* pairs of features. Co-ordinating features when the hands are holding the appropriate blocks (or constructions) means the fit will always be correct.

Once the Possible Reflect sub-goal has completed, there are two further types of production that exist for perceived correct constructions. First, if the construction is a layer (adults know the appearance of layers from the picture they are given), then a sub-goal is set to determine what to do with the layer (leave it if it is the first layer, or stack it if it is a subsequent layer). The size slots in the goal are then set to the next size and the goal is switched to Decide Search. Second, if the construction is not a layer, then the goal is immediately set to Decide Search, unless there are two pairs of constructions of the same size, in which case the Decide Fit goal is set. Both types of production can learn new declarative knowledge if it was not already known. For example, a correct pair could result in learning that blocks of the same size fit together, or that pegs fit in holes, if this knowledge is appropriate and was not already known.

When a construction is perceived as being incorrect, a sub-goal is set to disassemble the construction, and upon its completion the Decide Search goal is set. The block or construction that is removed is the last one that was fit.

The productions concerned with perceived correct constructions all have a success flag attached to them, and the productions concerned with perceived incorrect constructions all have a failure flag attached to them, so that the expected gain values of the productions can be updated.

5.4.4 The what, how, and where of learning in the model

Table 5.1 summarises what is learned in the model.

Table 5.1: A summary of what the model learns, and where it learns it.

Learning mechanism	What is learned	How it is learned	Where the learning occurs
Proceduralisation of declarative knowledge	Production rule in Decide Fit	The declarative knowledge that guided the fit is used to create a new production rule involving the features specified by the declarative knowledge	Fit Together
Reflection	Production rule in Decide Fit	Reflecting on a construction that is perceived to be correct means noticing that features can be co-ordinated when fitting. Therefore new production rules are created to search for two sets of features and fit by two sets of features	Assess Fit
Declarative knowledge	Declarative knowledge	Building constructions or analysing the blocks on the table can mean new knowledge is created	Scan Table Assess Fit
Success/failure flags	Expected gain value	Constructions perceived to be correct have their success statistic incremented, and constructions perceived to be a failure have their failure statistic incremented	Decide Fit Assess Fit

5.4.5 Model parameters

The model contains a variety of parameters that can be altered. Modifying each parameter should change the way the model behaves. The model does not use some of the mechanisms that exist in ACT-R, so not all ACT-R parameters are listed.

1. Decay rate.
2. Retrieval threshold.
3. Base level activation.

4. Default timing for production rules.
5. Noise placed on expected gain.
6. Block manipulation timing.
7. Saccade timing.
8. Fixation timing.
9. Removal of knowledge.
10. Insertion of knowledge.
11. Representation of the task.

5.5 Chapter summary

The Tower simulation and model have both been described in detail. Where appropriate, justification has been given for the selection of some of the mechanisms used by the simulation and the model. Some mechanisms, in particular the learning mechanisms incorporated into the model, are justified through a cycle of comparisons between the models' behaviour and the behaviour of adults and seven year olds. Therefore there are some remaining methods used by the simulation or the model that cannot be justified until model-data comparisons are made. The next chapter begins these justifications, where the behaviour of adults will be compared to the behaviour of the model.

6. Comparison of model behaviour with adult behaviour

Before attempting to make modifications to the model, the behaviour of the model must closely match that of adult subjects on a variety of measures. This chapter starts by explaining how the results from the model will be compared to the results of subjects, and what the model's parameters settings are. Comparisons between the model's behaviour and the adult subjects behaviour are then made on aggregate measures of temporal behaviour, observed behaviour, and within-task learning. These show the model to match the adult behaviour closely. This enables the examination of the model's behaviour to see what how and what the model learns. The chapter concludes by comparing the overt behaviour of a portion of one run of the model with one of the adult subjects. The model is able to suggest what cognitive processes subjects may be performing in between their overt behaviours on the task.

6.1 How to compare model behaviour with subject behaviour

There are no statistical guidelines for how to compare model and subject behaviour, because the type of comparison being done is the opposite of the type of comparison that standard statistical tests are designed for. The standard statistical test seeks to find out if two sets of data are reliably different from each other. When this is the case, some conclusions can be drawn based on the hypotheses being tested. When comparing the data from the model with data from subjects, the aim is to show that the model's behaviour is similar to the subject's behaviour. Statistical tests do not cater for this type of examination (see Grant, 1962). Although the statistical tests cannot be a proof of the model, when reliable differences occur between subject and model scores they can indicate where the model needs to be improved.

The models of development covered in Chapter 3 should serve as some guide to comparing subject and model scores on the nine principle measures of task behaviour reported in Chapter 4. None of the models use statistical tests; where comparisons are given, only the scores for the measures are reported. The model and subject scores on the nine Tower task measures will be reported in the same way, although model scores that are within 20% of subject scores will be highlighted. This serves *as a guide* to the measures for which the model provides a close match to subjects. An error of 20% may seem generous but has been used as model support in the past (e.g. Card, Moran & Newell, 1983, p.180). The only statistical tests that will be reported in comparisons between subject and model scores will be those where a reliable difference is found. A

reliable difference indicates when the score for the model does not match the score for subjects.

The measures of within-task behaviour (timing per layer and number of construction attempts per layer) will be reported in the same way as in Chapter 4. Correlations between model and subject layer scores provide an indication of whether the model scores follow the same layer trend as subject scores. In some cases, the scores may correlate and yet be inaccurate (e.g. the model scores could be twice as high as subject scores for each measure). To overcome this, the RMS error can be reported. The RMS error will measure the difference in scores between the model and subjects for each layer. For each layer, the difference between the model and subject scores is squared; all the scores are summed and then the root of this value is taken as the RMS error. The RMS error therefore indicates the extent of the mismatch between the model and subject layer scores. This will be reported as a percentage (i.e. the extent of the mismatch calculated as a percentage based on the subjects' average scores across layers).

6.2 Model and task simulation parameter settings

Table 6.1 shows the values of the model and task simulation parameters used in modelling the adult behaviour on the Tower.

Table 6.1: The values assigned to the model and task simulation parameters.

Model parameter	Value	Task simulation parameter	Value
Decay rate for DME's	0.05	Size of the fovea	Half width of smallest block
Retrieval threshold of DME's	0.0	Size of the parafovea	2.5 * Size of fovea
Base level activation of DME's	10.0	Probability of inaccuracy when fitting blocks together internally	0%
Default timing for production rules	50 ms	Probability of inaccuracy when internally determining whether anything obstructs a fit	0%
Noise placed on expected gain	0.04		
Block manipulation timing	550 ms		
Saccade timing	50 ms		
Fixation timing	200 ms		

A decay rate of 0.05 has been used in other ACT-R models where working memory decays (Lovett, Reder & Lebiere, 1997). A retrieval threshold of 0.0 is the standard setting for this parameter in ACT-R.

The base level activation parameter is set relatively high at 10.0 (the ACT-R default is 1.0). Timings associated with production rule firings increase as the activation of DME's in the conditions of the productions decrease. If a production which has several low activation DME's in its condition happens to fire, the production timing will be unrealistically high. Incorporating decay means that DME's of a low activation are likely to be part of productions that fire. The base level activation of DME's is set to 10.0 in order to null the effect of the DME's on the timing of a production. This means the timing for a production rule firing will generally be the same as the default production timing estimate.

The default production rule timing estimate of 50 ms is the standard setting for this parameter in ACT-R models. However, as stated above, this estimate does not rise based on activations of DME's. This is not in accordance with ACT-R theory, but a production timing estimate of 50 ms is used in the EPIC architecture (Kieras & Meyer, 1996), and 50 ms is the timing for a decision cycle in many Soar models (see Nelson, Lehman & John, 1994).

The expected gain noise parameter is set to 0.04. Such a low setting means that noise will only affect those strategies which have very similar expected gain values (those which differ in expected gain values by ≈0.2 or less). The parameter setting therefore only provides differences in strategy selection when strategies are approximately equal in expected gain value (i.e. when a run of the model begins).

The timing estimates for production rules which interact with the simulation differ from the default timing. Each interaction involving hand movements (picking up, dropping, rotating, fitting, turning over, and disassembling blocks) takes 550 ms. Interactions involving eye movements take 50 ms, and those involving fixations take (on average) 200 ms. Justification for these timings was given in Chapter 5.

For eye movements and fixations, some of the production rules involved must have a default timing of 0 ms so that the timing estimates for these actions can be adhered to. This is because in the implemented model, eye movements and fixations are accomplished by a sub-goal (for ease of modelling). The 0 ms timing of some production rules is therefore an implementation detail. The timing estimates for task simulation interactions are set in the model because it is within ACT-R that timings are made.

The size of the fovea and parafovea, as for eye movements and fixations, are based on estimates from a review of the vision literature (Baxter & Ritter, 1996).

The parameters associated with probabilities of inaccuracy are all set at 0%. The model is able to accurately accomplish all internal fitting (perceiving how two blocks will fit together without actually carrying out the fitting blocks together physically), and all internal obstruction checks (perceiving whether there is an object which obstructs two blocks being fit together). Adult subjects do not make constructions which are not flush on their outer edges, and do not attempt constructions when objects impede the fit.

6.3 Aggregate model behaviour versus aggregate adult behaviour

The model and the task simulation include elements which are random in nature. This is so that one element can be selected from several, when all are equal. For example, in the model, when there are several instantiations of the same rule, one will be selected at random. In the task simulation, when scanning the table, the model looks at a random block from all of those that have not been fixated on.

In order to try and remove the influence of random aspects of the model and task simulation behaviour, the model was run ten times to match with the five adult subjects. Ideally, the model would be run as many times as possible. A limiting factor is that, when modifications to the model are made, they need to be run the same amount of times as the original model (so that comparisons between models can be made). Each single run of the model is time consuming, so ten runs of the model are used throughout this thesis when aggregating the model's performance.

The ten runs of the model used the parameter settings detailed above, with the same starting state for the blocks (shown in Figure 5.1 in Chapter 5; this is the same starting state as for the adult subjects). The data from the model is aggregated over the ten runs, and compared with aggregated data from the five adults who completed the Tower.

The model and subject data will be compared in three ways. First, a comparison between model scores and subject scores is given for the nine measures outlined in Chapter 4. Second, the strategy behaviour will be compared to check that the model has the same distribution of strategies that subjects have. Third, within-task timings and construction attempts are compared.

6.3.1 Overall behaviour

Table 6.2 shows the scores for adult subjects and the model for the nine measures of overall behaviour outlined in Chapter 4. The model is within a twenty-percent range on seven of the nine measures. On the two particularly important measures (number of construction attempts and time taken) the model is within two percent of subjects (101.3% for construction attempts; 101.9% for time taken). The model provides a good match to the subject data.

Table 6.2: Summary of scores on the principle measures of behaviour on the Tower task, for adult subjects and the ACT-R model. Model scores within 20% of subject scores are indicated with a *. Standard deviations are indicated in parentheses.

	Adults	Model
Construction attempts[4]	22.8 (2.9)	23.1* (2.4)
Time taken	126.6 s (34.0)	129.0 s* (31.5)
Incorrect constructions involving blocks of the same size	1.8 (1.1)	3.0 (2.4)
Incorrect constructions involving blocks of different sizes	0.6 (0.9)	0.0
Average number of times the same blocks are fit together in the same way again	0.0	0.0*
Average number of times the same blocks are fit together in different ways	1.3 (0.6)	1.3* (0.5)
Time between correct constructions	6.0 s (6.4)	6.4 s* (7.2)
Time between incorrect constructions	5.0 s (6.9)	4.6 s* (6.9)
Number of task appropriate features	1.9 (1.0)	1.8* (0.4)

6.3.2 Strategy behaviour

Figure 6a shows the strategy distributions for producing a layer for the adult subjects and the model. The model does not match the frequency data but has the same distribution of strategy type (there are no reliable differences between the model and adults for any of the strategy types but this is probably due to noise).

---

[4] The number of correct constructions produced can be more than twenty (the optimal amount).

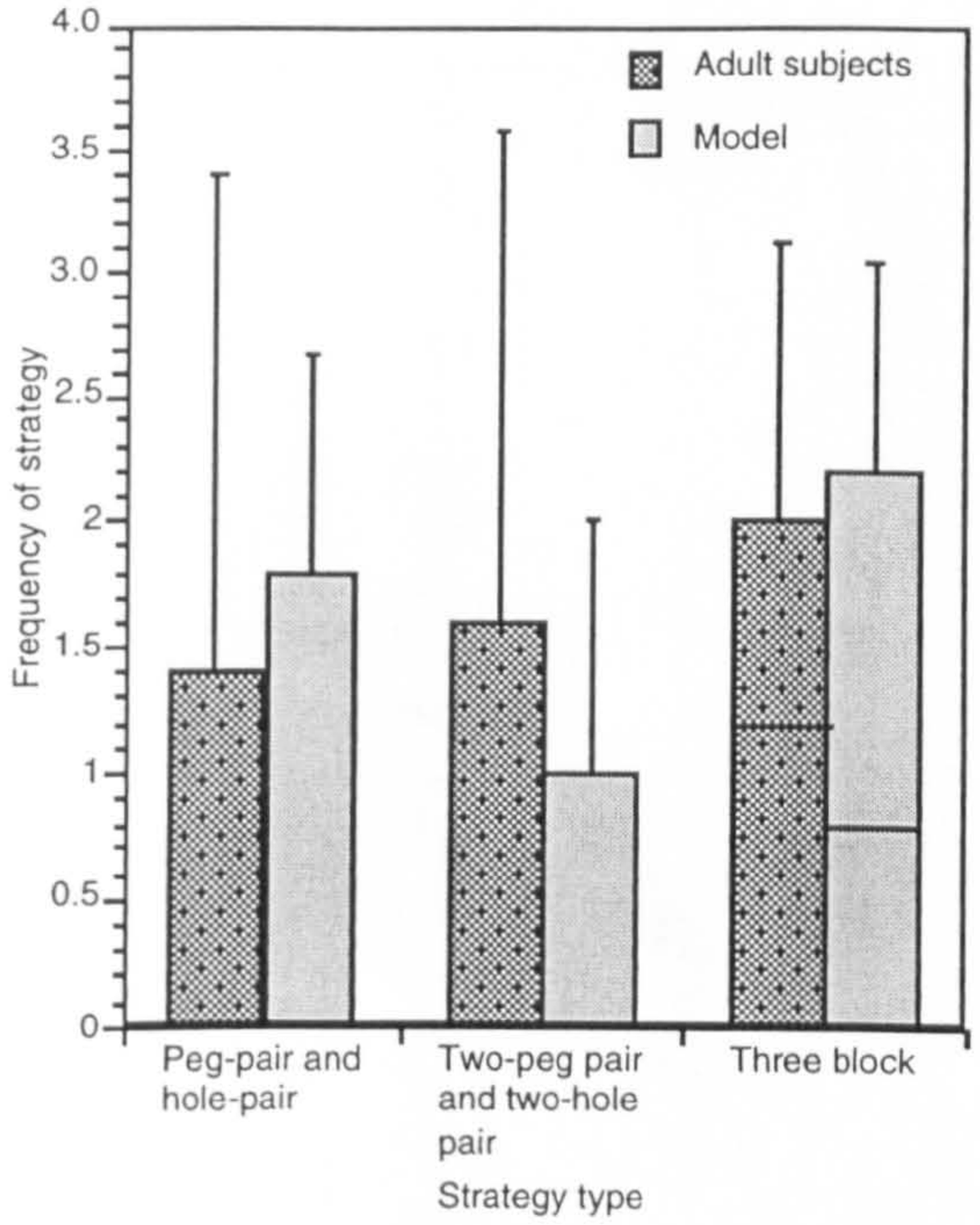


Figure 6a: The distribution of strategies for the adult subjects and the model.

Figure 6a suggests that the model does not use the two-peg and two-hole strategy as often as adults. Breaking down the three block strategy to find which type of initial construction pair is made is revealing. Figure 6a shows this using a horizontal bar, with the number of initial peg or hole-pairs being below the line, and the number of initial two-peg and two-hole pairs being above the line. Including these figures with the initial pairs of the other strategies, the comparison is very favourable (both model and subjects produce 2.6 initial peg-pair or hole-pair constructions, and 2.4 initial two-peg or two-hole constructions). The model prefers to add another single block to a pair construction once a two-peg or two-hole construction has been produced.

The type of incorrect constructions created by subjects and the model is shown in Figure 6b. The subjects produce incorrect constructions that the model does not (e.g. peg in hole), but these occur in small frequencies.

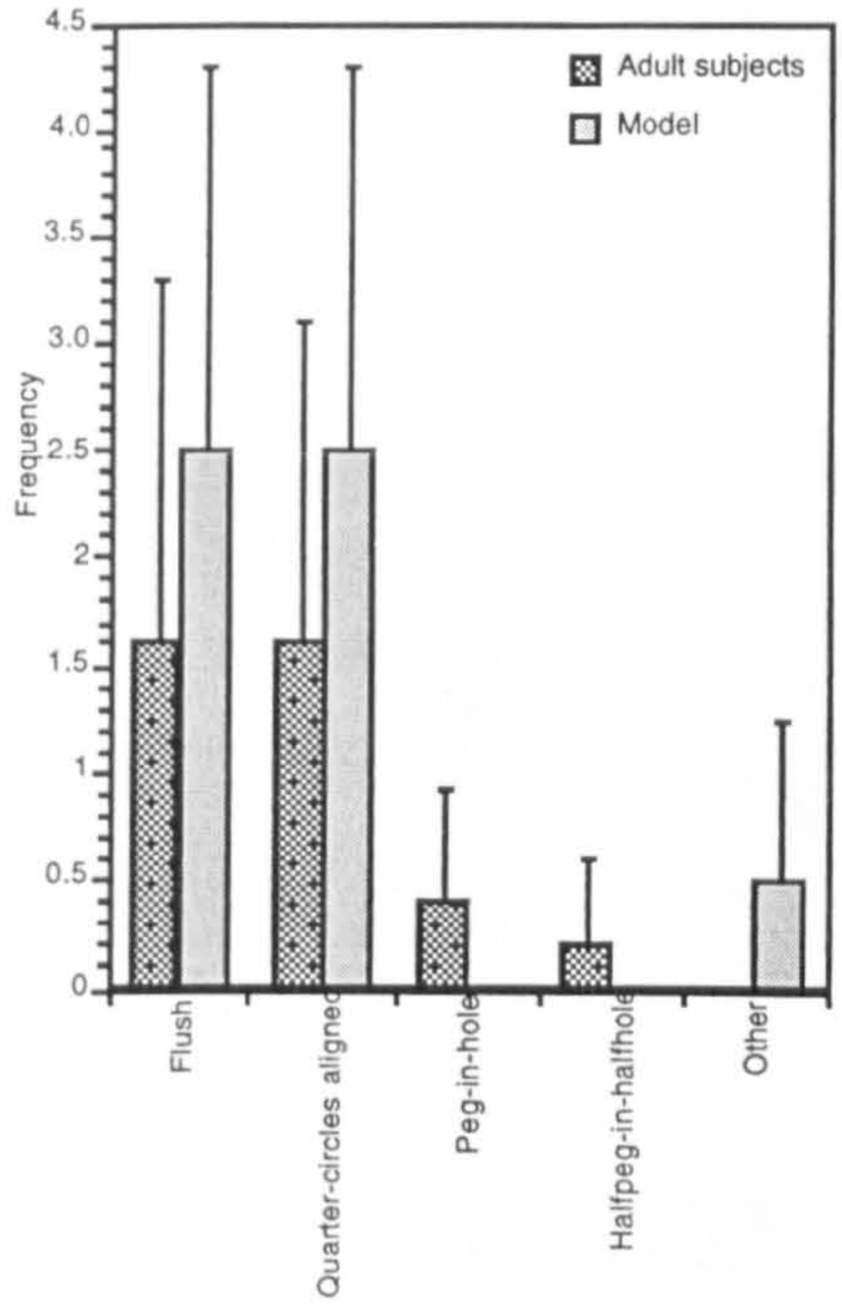


Figure 6b: Frequency distributions for attributes of incorrect constructions.

6.3.3 Within-task behaviour

There is a reduction in time taken to produce subsequent layers. A model–subject comparison can be seen in Figure 6c. The layer timings have a correlation of 0.96 and a RMS error of 4.1% (the RMS error measures the average percentage difference in timings between the model and subjects for each layer). The error bars in the figure (subjects to the left, model to the right) show that in general, the variance in the layer timings of the model tend to be slightly higher than those of the subjects.

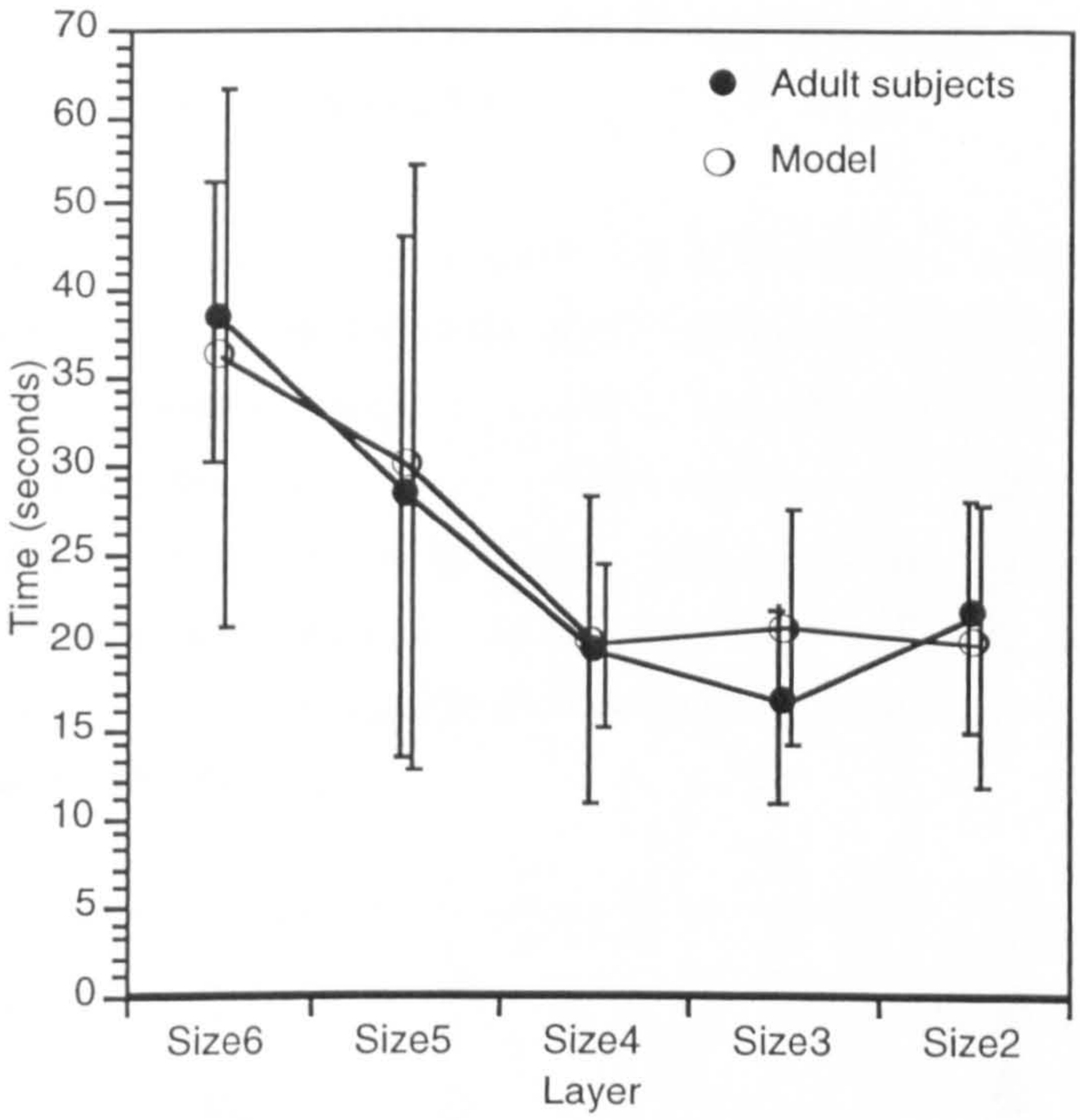


Figure 6c: Time taken (in seconds) for adult subjects and the model to complete each layer.

Model–subject comparisons for the number of construction attempts produced each layer are given in Figure 6d. The model's layer construction attempts do not correlate with those of subjects (r=0.40; RMS Error=5.7%). This is not surprising because the construction attempts for adults form a near flat curve.

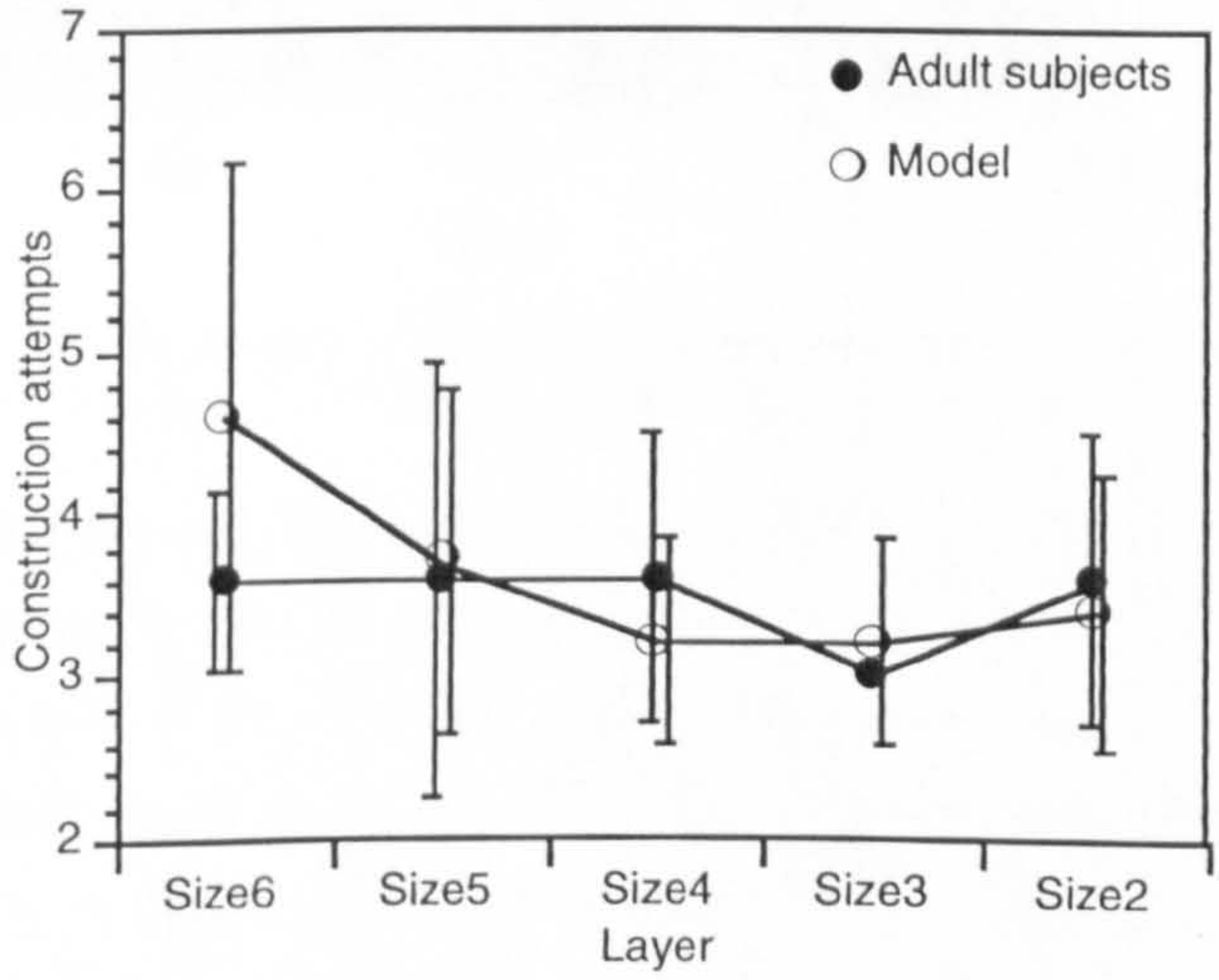


Figure 6d: Number of construction attempts made by adult subjects and the model in completing each layer.

This flatness of the adult construction attempts also explains the poor correlation between adult timings per layer and adult construction attempts per layer (r=0.53). The model timings per layer correlate well with the model construction attempts per layer

(r=0.95). Timings and construction attempts should correlate well because an additional construction attempt will take additional time.

The lack of a reduction in construction attempts for adults suggests that any task learning by adults is unobservable. That is, the reduction in the time to produce subsequent layers is because there are less blocks to select from as the task progresses (i.e. the reduction is due to reduced visual search). The behaviour of the model can be analysed in more detail to find where the time is spent when constructing each layer. Having timings for all production firings in the model allows the extraction of timings for moving and fixating the eye, manipulating the blocks, and cognizing. The time separations for each of these processes can be seen in Figure 6e.

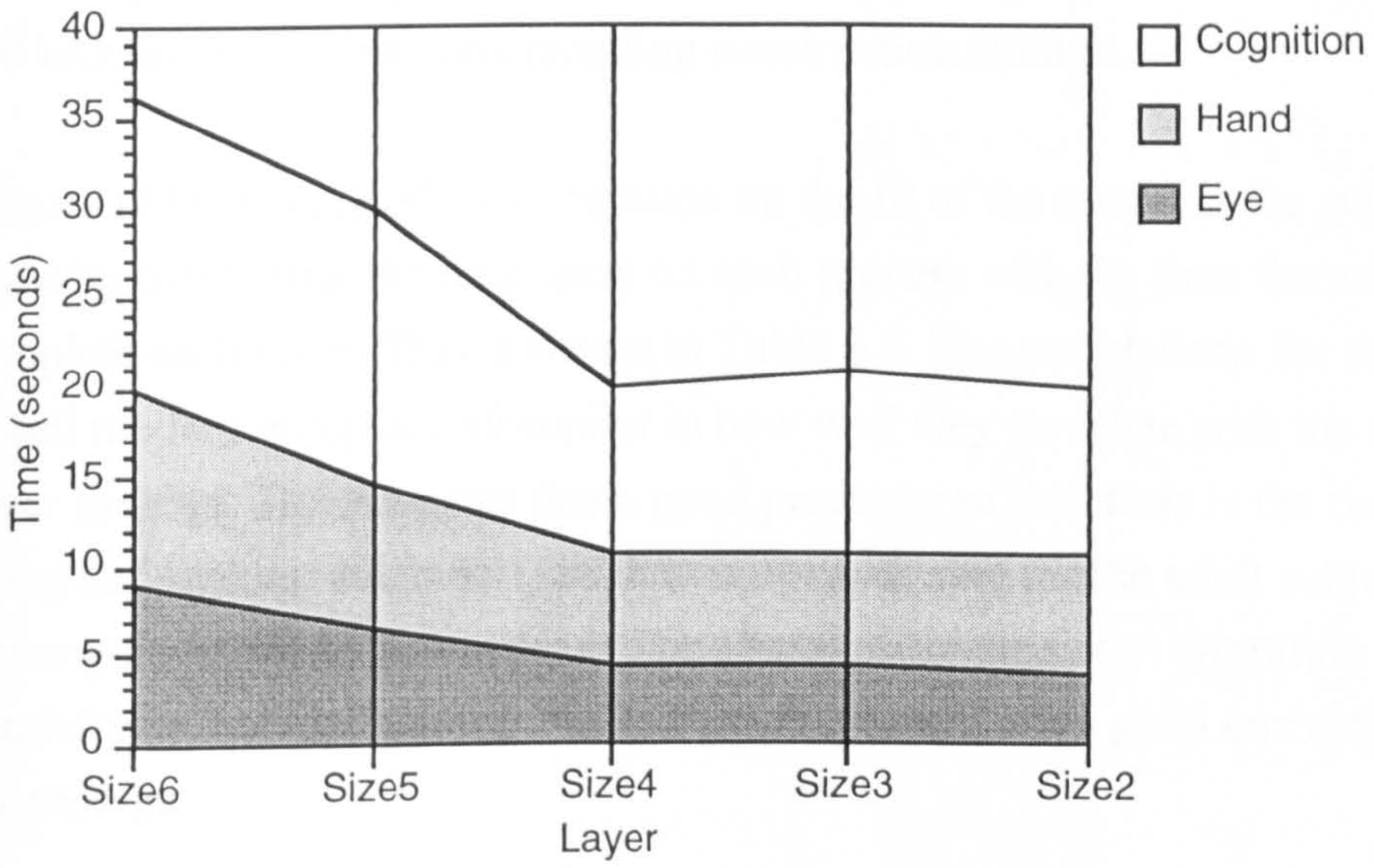


Figure 6e: Contributions (in terms of time) of cognition, eye movements and fixations, and hand movements involved in completing each layer of the Tower.

The model does not incorporate learning in its simulation eye and simulation hands. The timings for the eye and hands are fixed across the whole task. The reduction in the amount of eye and hand use between the first and second layers constructed (size6 and size5) accounts for 84% of the total reduction in time taken between those layers. The eye and hands account for 24% of the reduction in times for the second and third layers constructed (size5 and size4). A reduction in cognitive effort therefore accounts for 16% and 76% of the reduction in time taken between the first and second, and second and third layers respectively. The reduction in time between the first, second, and third layers is due to cognitive learning as well as a reduction in visual search. This would suggest that adults are learning on the task even though it is not reflected in their construction

attempts. The timings for the eye and hands remain constant after the size4 layer is produced, suggesting a minimum time for searching and constructing.

Each of the three individual processes (eye, hand, and cognition) correlate very well with the full model for timings per layer (minimum r=0.97). The eye and hand timings per layer also correlate very well with the model's construction attempts per layer (r=0.97 and r=1.00 respectively), although the cognitive timings are not as high (r=0.87). The number of construction attempts made influences the eye and hand timings more than cognition timings. This may be because the internal fit procedure causes some fit attempts to be aborted before a physical construction attempt is made. This represents cognitive effort without the observed behaviour of attempting to fit blocks. To some extent this may explain why adult timings and construction attempts do not correlate well, because timing will increase without a corresponding construction attempt.

The influences of the eye, hand, and cognition on the fit of the model to the subjects can be examined by correlating the time spent on each process with the time the adult subjects take to complete each layer. This is shown in Table 6.5. The correlations for the eye, hand, and full model timings are all similar in how well they correlate with the adult subject layer timings. This suggests that a good predictor of task time is the time spent manipulating and looking at blocks (this data is not available for the adult subjects). The timings for cognition would also appear to be a reasonable predictor: cognition accounts for substantially more time than the eye and hands, yet still has a good correlation with adult layer timings.

The time spent interacting with the simulation eye and hands should also correlate well with the number of construction attempts that adults produce, because any interaction is toward the goal of completing each layer. Further construction attempts should mean more fixations on blocks, and more blocks being manipulated by the hands. Table 6.5 shows that this is not the case (remember the construction attempts for the adults did not correlate well with the adult layer timings either). However, the adult construction attempts are flat across layers (as stated above).

Table 6.5: Correlations between adult subjects and the model when individual model processes are extracted out of the timing data.

Model process	Process time	Correlation with subject layer timings	Correlation with subject layer construction attempts
Full model (excluding stacking final top block)	126.0 s	0.96	0.40
Eye only	27.5 s	0.97	0.35
Hand only	38.5 s	0.98	0.37
Cognition only	60.0 s	0.91	0.32

The timing for interactions with the task simulation hands is 38.5 seconds. This provides further evidence regarding the discussion in Chapter 4 as to whether the increase in task time for seven year olds is simply because they make more construction attempts. The *total* time that the model spends manipulating blocks only accounts for 81% of the *extra* time that seven year old children take in building the Tower. The children would have to spend more than twice as long as adults do in manipulating blocks.

In the model, the time spent moving and fixating the simulation eye, and manipulating the simulation hands, accounts for 51% of the total task time. This suggests that any cognitive model of a task involving interaction which does not have a task simulation, will over-estimate the time spent on cognition and therefore may arrive at inappropriate conclusions.

6.4 What does the model learn?

Learning in the model centres upon productions. New productions are learnt, and all productions have an associated expected gain value which fluctuates based on the number of times the production has led to success, the number of times the production has led to failure, and the time taken to achieve the success or failure. The main component of the model to influence how constructions are made is Decide Fit. Examining when new productions are created in Decide Fit, and how their expected gain values change within the task, will show what the model learns whilst completing the Tower.

The expected gain values were recorded every time the Decide Fit goal was the focus of the model. Recording the values every cycle is unnecessary because the values will only change when a fit attempt succeeds or fails. In the graphs depicting expected gain values, the increment in the cycle interval is not regular because the recordings were only made when the Decide Fit goal was the focus. The graphs show the expected gain values from one sample run of the model.

Figure 6f shows the decline of the expected gain values of the standard fit productions (i.e. the general fitting productions that the model begins with). The two general fitting productions (detailed in Chapter 5) are not shown because they never fire.

Every dip or rise in the timeline of a production represents a time when the production was used either successfully or unsuccessfully (including unsuccessful fit attempts which were aborted by internal fit or internal obstruction checks). A change in the expected gain value represents the success or failure of a production in achieving a correct construction, and the cost of achieving it (in terms of the time taken). This means the value could decline even on success, if that success took a long time. Note that it is possible that an initial success for any of the productions will not be seen, because the expected gain values begin at their peak.

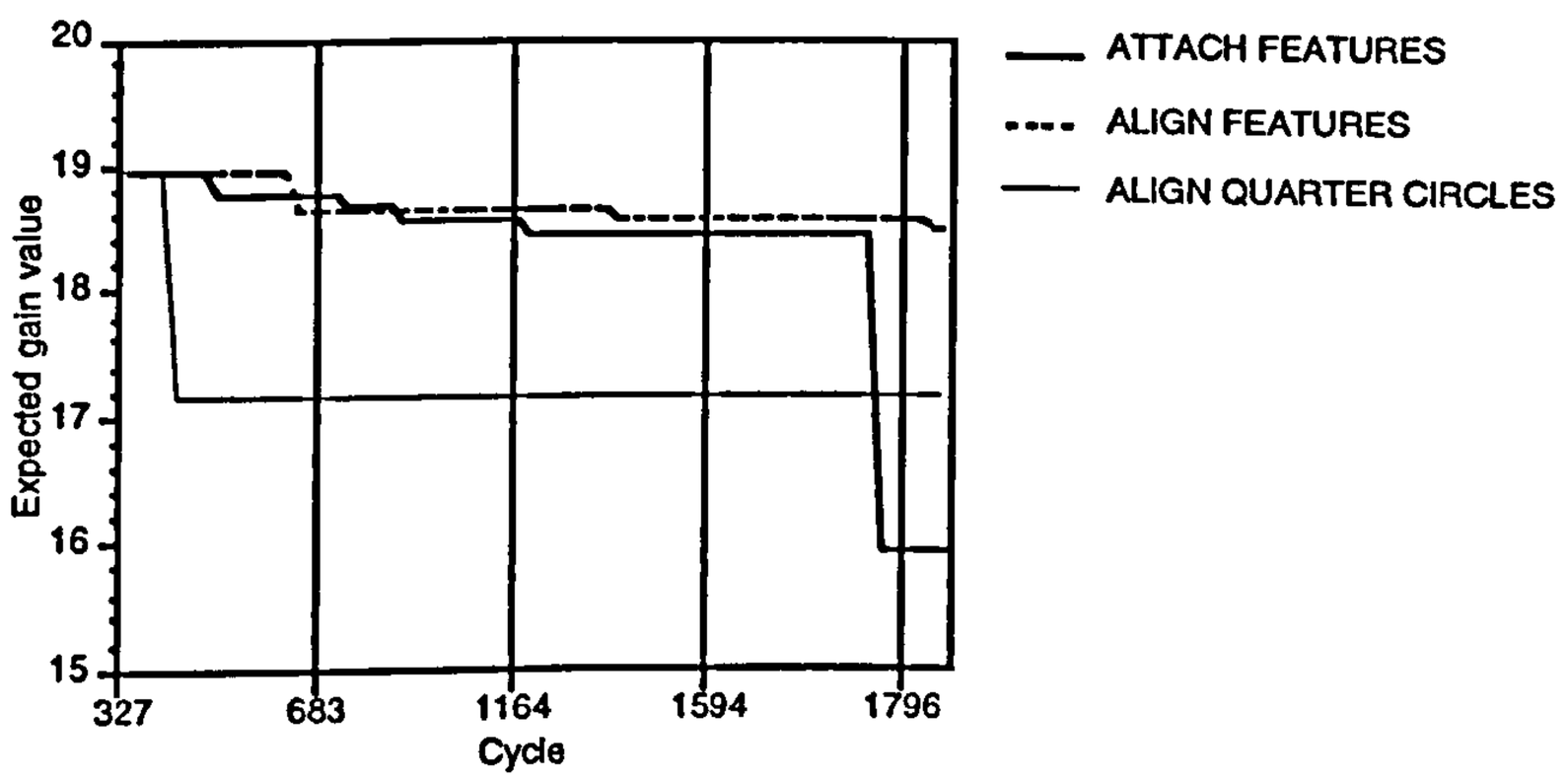


Figure 6f: The expected gain values for the standard Decide Fit productions during the course of a sample run of the model.

The expected gain value for the production which aligns features does not decline as much as the expected gain values for the attach and quarter-circle productions. This is to be expected given that aligning halfholes or halfpegs yields a much better chance of success (see Chapter 4). The changes in expected gain values reflect learning in the task – when productions are misapplied (and therefore result in an error of some sort) the value

decreases. The attach production and the quarter circle production both suffer single significant decreases in their expected gain values. For the attach production this occurs because the internal fit check results in the fit being aborted, and for the quarter circle production it is because an incorrect construction is made. The drop in value for the quarter circle production is bigger than the drop for the attach production because more time is spent in reaching the failure.

During the run, several new productions are created. Figure 6g shows how the expected gain values fluctuate for newly created productions. For clarity, the productions are grouped into the standard productions (which the model starts out with), the new productions that are created from the proceduralisation of declarative knowledge, and the new productions created by co-ordinating features. One of the new productions created from proceduralising declarative knowledge can be seen in Appendix E.

The initial sharp rise in expected gain value for the new productions reflects when the first of that production type was created. As may be expected, the expected gain values for the co-ordination productions remain close to their peak. This is because they should always result in success. Near the end of the run, the productions resulting from proceduralised knowledge have lower expected gain values than the co-ordination productions. This is because they do not always contribute to the creation of a correct construction (the productions are instantiations of the standard productions, for a specific pair of features; they do not co-ordinate features). However, at the very end of the run, one of the proceduralised productions fires and results in a success which boosts its expected gain value.

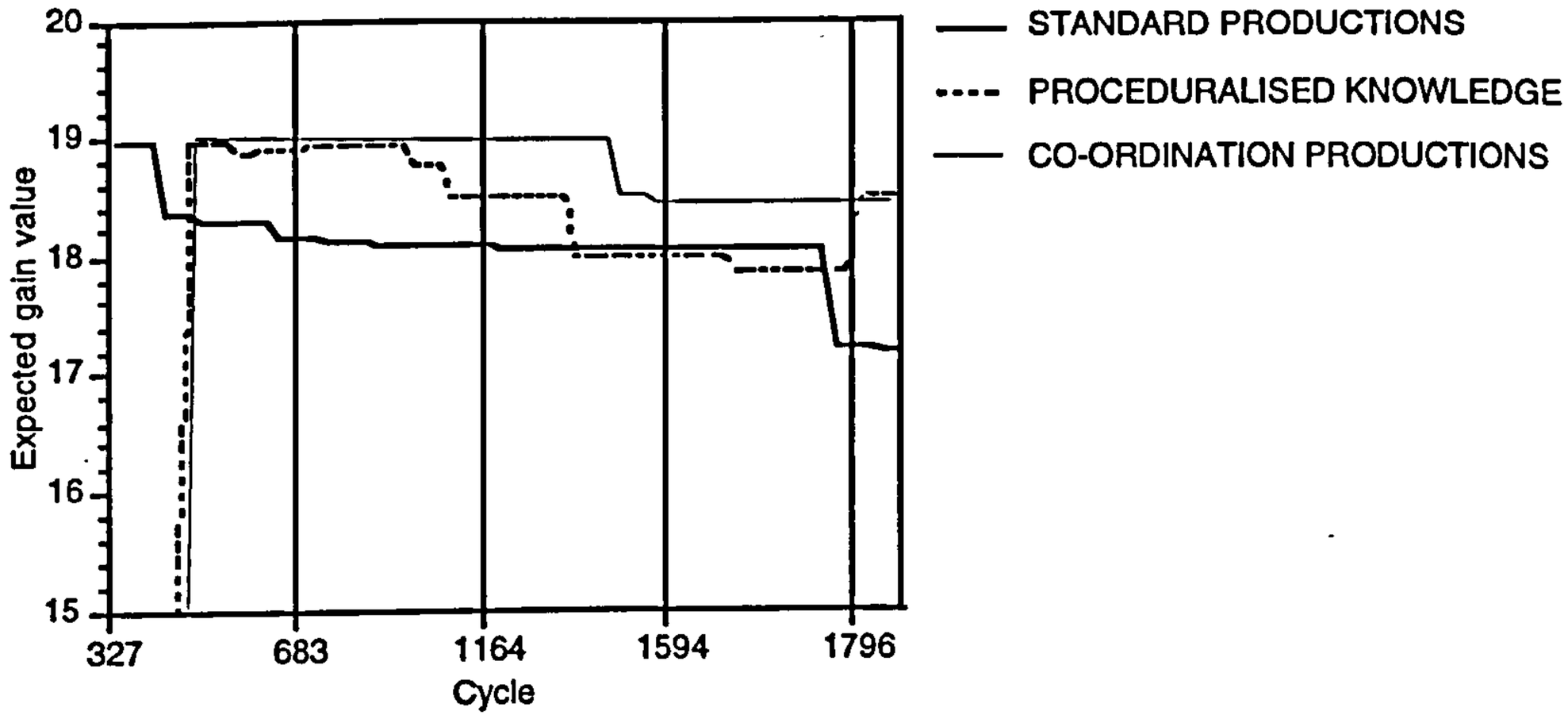


Figure 6g: The expected gain values for the three types of fitting productions in the Decide Fit module during the course of a sample run of the model.

The selection of which production to fire when the Decide Fit goal is the focus should mirror the expected gain values at that point in time (although there is expected gain noise of 0.04). However, this is complicated by the blocks that are being held at that time, because new productions are for specific features: if the blocks do not have those features, they cannot fire. Figure 6h shows the number of times each of the three types of fitting productions fire during the sample run. The co-ordination productions fire later in the task, yet they are only used three times for this run. This is mainly because they co-ordinate two features – if the selected blocks do not have those features, the productions cannot be matched.

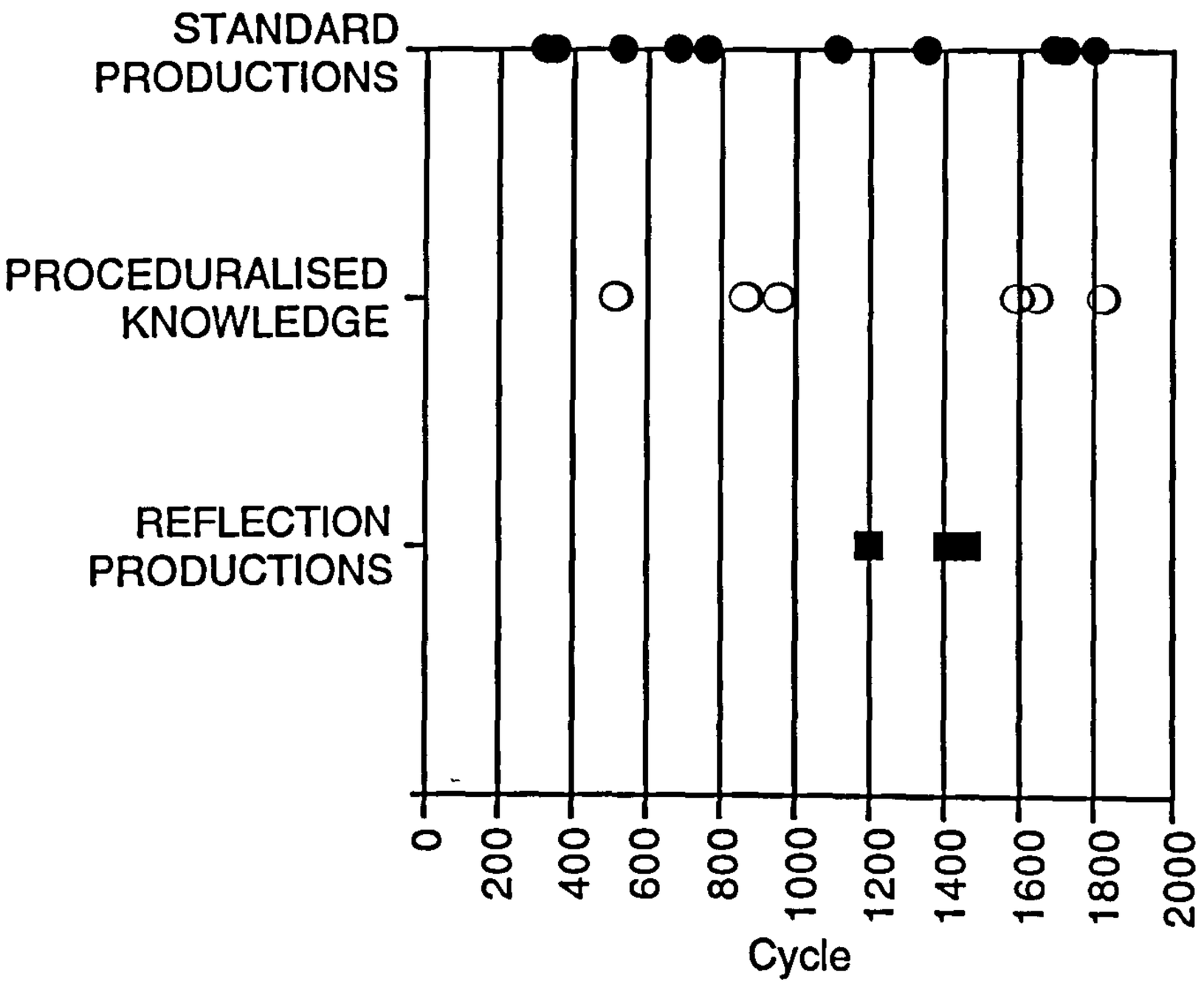


Figure 6h: The distribution of production firings for each of the three types of production.

The model therefore learns both new productions and the expected gain values associated with them. This does not always ensure that the most effective productions will be used because this is dependent upon which blocks are selected.

6.5 Comparison of the model with an individual subject

To give the reader some sense of the model running, a small subset of a sample run of the model is compared with that of one of the adult subjects (the sample run presented is chosen because it fits well to the particular subject). This is shown in Table 6.6. The first portion details the first construction attempts by both the subject and the model. This shows both making the same incorrect construction, with both immediately recovering from this error. At time 00:00:22, the subject makes the same incorrect construction as the model does at time 00:00:12. The next construction code for both the subject and the model is to disassemble the incorrect construction.

The second portion details behaviour for the model and subject in the middle of the task, where both achieve error-free construction of the third largest layer. The model behaviour includes more codes because it allows the recording of every possible construction behaviour, whereas only the observed behaviour for the subject is available.

Table 6.6: A comparison of subject behaviour with the behaviour of the model, for both incorrect and correct constructions.

Incorrect subject behaviour	Incorrect model behaviour
00:00:20 Considers-incorrect/flush/qc-aligned (Size6HalfPeg-Peg,Size6HalfHole-Hole) 00:00:22 Incorrect/flush/qc-aligned (Size6HalfPeg-Peg,Size6HalfHole-Hole) 00:00:27 Disassembles-incorrect (Incorrect[Size6HalfPeg-Peg,Size6HalfHole-Hole]) 00:00:30 Correct(Size6HalfHole-Hole,Size6HalfHole-Peg)	00:00:11 Considers-incorrect/flush/qc-aligned (Size6HalfPeg-Peg,Size6HalfHole-Hole) 00:00:12 Incorrect/flush/qc-aligned (Size6HalfPeg-Peg,Size6HalfHole-Hole) 00:00:13 Disassembles-incorrect (Incorrect[Size6HalfPeg-Peg,Size6HalfHole-Hole]) 00:00:15 Considers-correct(Size6HalfHole-Peg,Size6HalfPeg-Peg) 00:00:16 Correct(Size6HalfHole-Peg,Size6HalfPeg-Peg)
Correct subject behaviour	**Correct model behaviour**
00:01:16 Correct(Size4HalfPeg-Peg,Size4HalfPeg-Hole) 00:01:18 Correct(Size4HalfHole-Hole,Size4HalfHole-Peg) 00:01:20 Correct(Size4PegPair,Size4HolePair) 00:01:27 Correct-qc-d-in-qc-e(Size4Layer,Size5Pyramid)	00:00:53 Considers-correct(Size4HalfPeg-Hole,Size4HalfPeg-Peg) 00:00:55 Correct(Size4HalfPeg-Peg,Size4HalfPeg-Hole) 00:00:58 Considers-correct(Size4HalfHole-Hole,Size4HalfHole-Peg) 00:01:00 Correct(Size4HalfHole-Hole,Size4HalfHole-Peg) 00:01:04 Considers-correct(Size4PegPair,Size4HolePair) 00:01:06 Correct(Size4PegPair,Size4HolePair) 00:01:07 Correct-qc-d-in-qc-e(Size4Layer,Size5Pyramid)

In the first portion of the run, there are two points at which the model decides how to fit blocks together. The first occurs when the incorrect construction is selected. All expected gain values will be equal at this point because it is the first time that a construction is made. The second occurs when the correct construction is made. At this point, the expected gain value of the align-quarter-circles production has been reduced to 17.10 whilst the other productions remain at their peak of 18.95. This is because the production resulted in an incorrect construction.

The second portion of the run is more interesting because it is taken from behaviour halfway into the task. At this point the expected gain values of productions will have fluctuated based on the successes and failures that have occurred previously in the run. Table 6.7 shows the expected gain values for all productions that are in competition when deciding how to fit the two size four blocks that have been selected (the two blocks are shown alongside time 00:00:53 in the second model portion of Table 6.6). The list of productions represents only five out of a possible eleven, because the blocks selected do

not permit the other six to fire. All three of the standard productions have dropped to a low enough level of expected gain that realistically the only two productions in competition are those that have recently been created. The alignment production is selected.

Table 6.7: The expected gain values for productions in competition before the first construction in portion two of Table 6.6 is produced.

Production	Expected gain value
DECIDE-FIT-FROM-PRE-KNOWN-ATTACHED-FTR	17.17
DECIDE-FIT-FROM-PRE-KNOWN-ALIGNED-FTR	18.63
DECIDE-FIT-FROM-PRE-KNOWN-ALIGNED-FTR-QC	17.13
NEW-DECIDE-FIT-ALIGNMENT-PRODUCTION286	18.95
NEW-DECIDE-FIT-COORD-PRODUCTION290	18.95

The model trace from the sample run can be examined. The trace shows what is happening in the model (in terms of production firings) between each observable construction behaviour. Figure 6i shows the trace from the creation of the first construction to the creation of the second (from 00:00:11 to 00:00:16 in Table 6.6). On this occasion, the disassembly arises from an attempt to fit the two blocks together but not being able to do so because halfpegs face each other. This is indicated by the "NO-FIT" production.

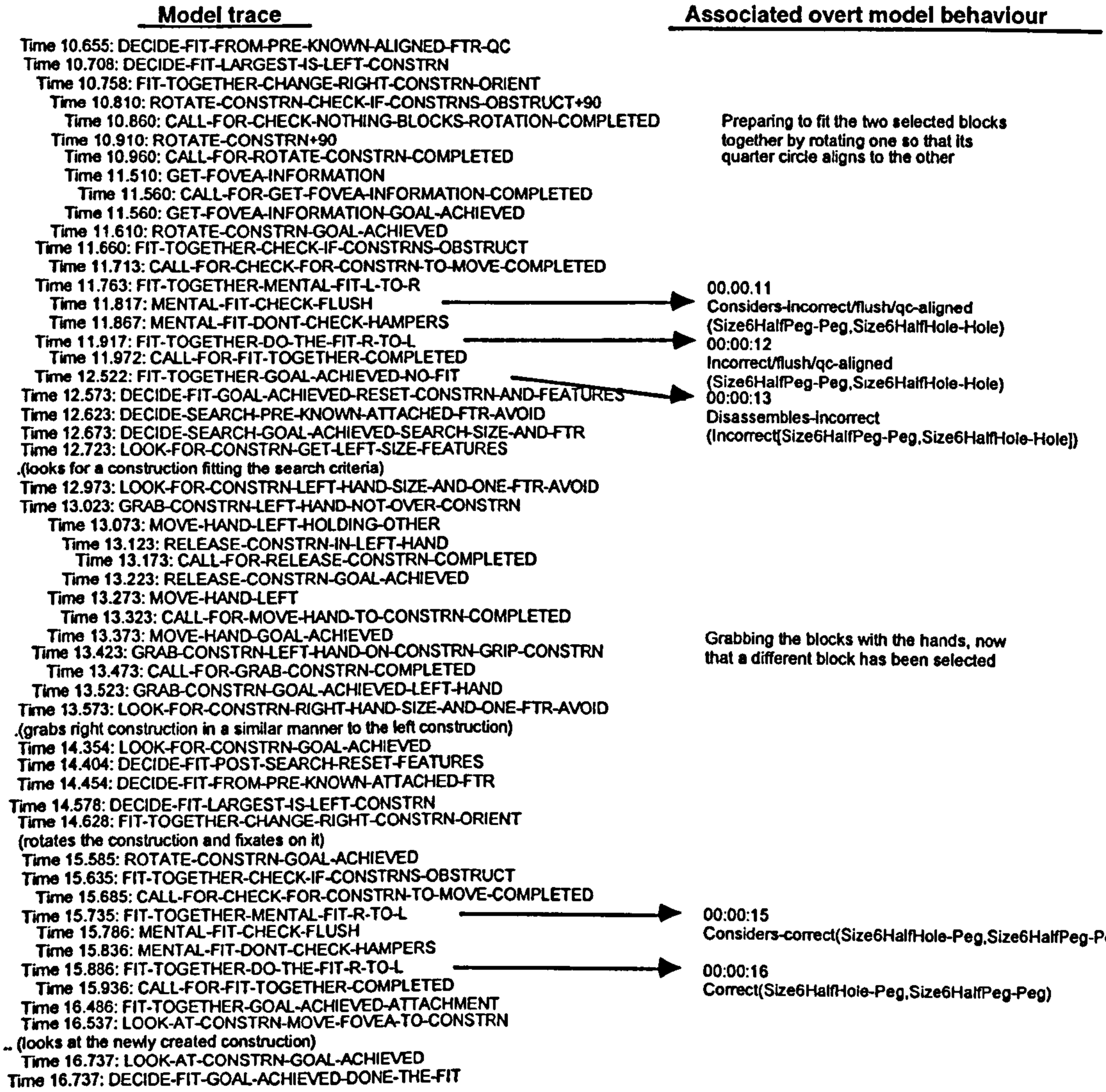


Figure 6i: A portion of the trace of the model.

Sub-goals are indicated in the trace by an indent before the production name. The trace begins after two blocks have been selected and are held in each hand. The model decides to fit the blocks by aligning their quarter circles (time 10.655), and so rotates one of the blocks so that the quarter circles align to each other (time 10.910). Before fitting the blocks, the internal fit process is carried out to check that the intended fit will be flush on its outer edges (time 11.817). This check is carried out successfully. However, the model decides not to check if any objects obstruct the fit (time 11.867). This turns out to be the case, so when the model tries to fit the two blocks, it finds they cannot be fit together (time 12.522). This results in an incorrect construction that is immediately disassembled (for both subjects and the model, a construction attempt is involved whenever two blocks

are brought together such that they touch each other; see the codings section of Chapter 4).

Control is then returned to the Decide Fit module, which flags a failure and decides to reset both the constructions and the features (time 12.573; it could have decided to try fitting by different features). This means that at least one of the blocks will be discarded. Some of the Decide Search and Look productions indicate this by having a "-AVOID" tagged on to the production name (e.g. time 12.623). This will avoid selecting the same two blocks again. The result is a new block being selected for the left hand, and one of the old blocks being selected for the right hand.

The Decide Fit module now decides to fit the blocks by attaching features (time 14.454; the peg and the hole, because these are the only block features that will attach). A similar procedure to the previous one now follows (time 14.628–15.836): one block is rotated so that the peg of one block faces the hole of the other; the internal fit process checks that the intended construction will have flush outer edges; the model decides not to check for objects obstructing the fit. The fit is then completed (successfully this time, at time 15.886). The model then needs to examine the new construction so that it has a representation of the construction and its features (time 16.537).

6.6 Summary

The model provides a good fit to the adult subjects on seven of the nine principle measures outlined in Chapter 4. The behaviour of the model has been examined in detail so that the reader has some understanding of how the model is able to complete the task. This includes learning within the task and how this affects behaviour. The model is now of a sufficient standard that theoretically motivated modifications can be made in order to examine their influence on development.

7. Individual modifications to the model

Chapters 2 and 3 outlined the need to test theoretical mechanisms of development within a computational model. Chapter 4 outlined a developmental task which could be used to develop a computational model. Chapters 5 and 6 outlined a model and a simulation of the developmental task, and showed that the model provided a good match to the subject data.

This chapter examines the influence that theoretical mechanisms of development have on the behaviour of the model. First, the fit between the existing model and seven year old children is examined. Second, each developmental mechanism is taken in turn, and implemented within the computational model. The behaviour of each modified model is compared to the behaviour of seven year old children on the task. Third, a summary of the modifications is given.

7.1 Fit between the behaviour of the original model and the behaviour of seven year old's

The fit between the original model and the seven year old children should first be examined before making modifications to the model. This will enable identification of those measures which modifications have moved closer to the seven year old behaviour. Figure 7a shows the percentage fit of the model to both the adult subjects and the seven year old subjects. The percentage is calculated by dividing the subject scores by the model scores, such that percentages below 100% indicate the model under-estimating subject scores, and percentages above 100% indicate the model over-estimating subject scores.

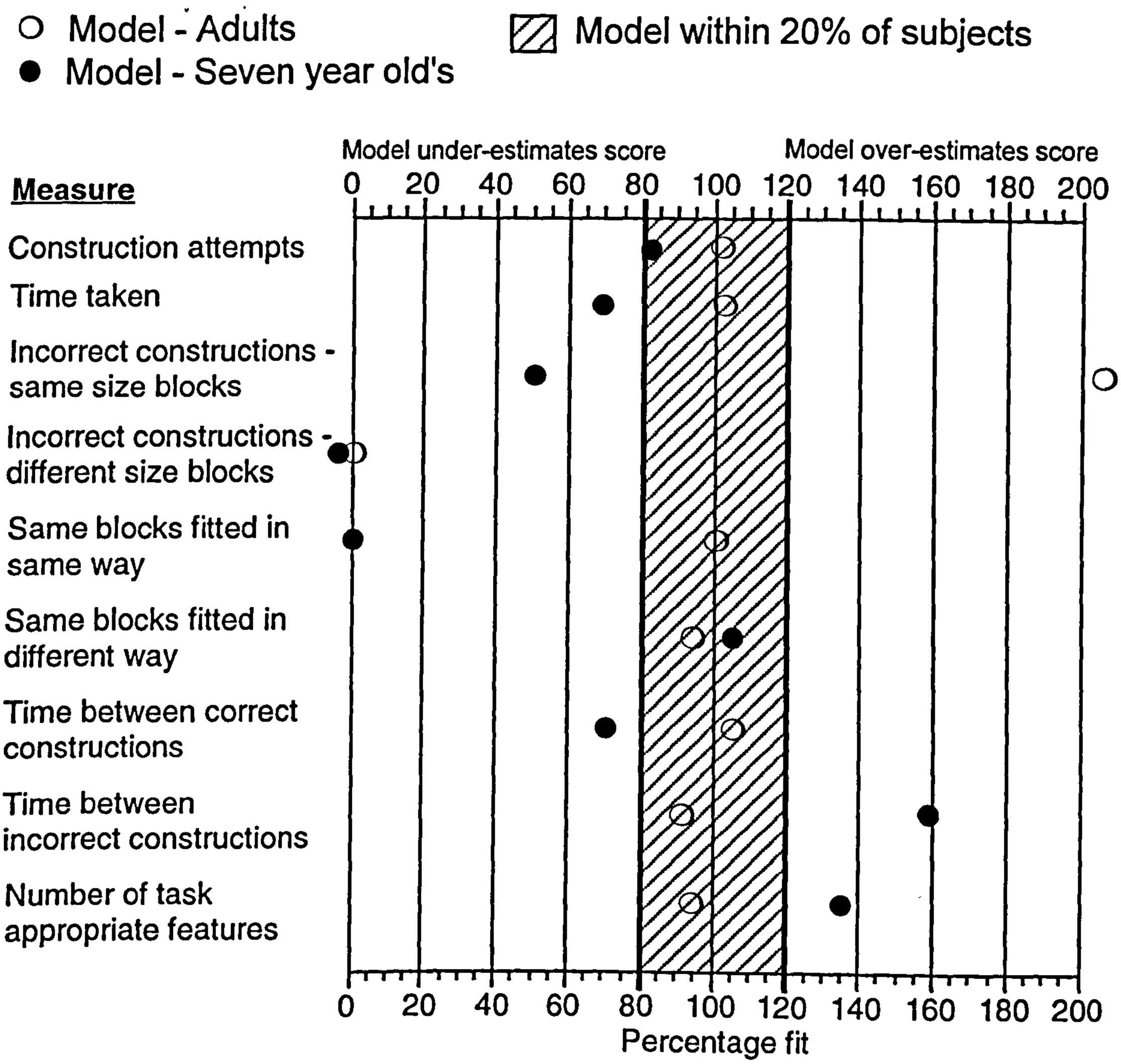


Figure 7a: Percentage ratio between model scores and adult scores, and model scores and seven year old scores, for measures of behaviour.

As can be expected, the standard model under-predicts the seven year old scores on most measures. There are only two measures that are within 20% of the scores of the seven year old subjects. The model *over-estimates* for the time between incorrect constructions because the incorrect constructions produced by adults bear great resemblance to correct constructions and therefore take longer to disassemble; the children's incorrect constructions do not (see Chapter 4). Figure 7a shows that the behaviour of the original model does not provide a good fit to the behaviour of seven year olds.

7.2 Modifications to the original model

When making modifications to the model, a high degree of confidence is required that a modification actually maps on to the aspects of development that theories propose. One problem with this is that some theories use different terminology to describe very similar theoretical concepts (see Chapter 2). This makes the job of mapping a modification to a concept more difficult than it should be. This means that some modifications may not map correctly onto the theoretical concept they aim to examine, because the theory has been vague in its definition of the concept.

All mechanisms are going to be implemented within a *computational framework.* Each mechanism has to be implemented in the model in a way which most effectively maps onto the theoretical description of the mechanism. The job of the modeller is to ensure this is the case. For each mechanism, a full description of how the model was modified (i.e. how the mechanism was operationalised in the model) will be given. The reader should be aware that the modifications to the model, by necessity, represent a computational approximation of the theoretical mechanisms.

The modifications made to the model are mutually exclusive; one parameter or one part of the model (e.g. expected gain noise) is changed, with all other aspects of the model remaining unaltered. Several modifications are presented. Some modifications will involve examining *different values* of a single parameter (e.g. varying retrieval threshold from one to nine in steps of one unit). For every modification and level of modification that is detailed, the model is run ten times (as it was when comparing the model's behaviour with adult behaviour). This data is then aggregated and compared to the behaviour of seven year old's on the task.

The modifications that involve examining different values of a single parameter are not examined at micro-levels of the parameter (e.g. a parameter will be examined in steps of one unit rather than at some lower-level). There are two reasons why no effort is made to examine micro-levels of parameters. First, the chapter is concerned with showing that modifying architectures is a useful method to examine development, and an exact match on all measures is not necessary for this. Second, an exact fit on most measures is only expected when more than one modification is made simultaneously to the model.

Comparisons between the modified models and the seven year old data cannot be reported for every measure due to space limitations. Instead, for every modification, predictions are made as to which measures should change. These are then analysed. The fit between the modified models and the seven year olds for all measures is summarised at the end of the chapter.

Statistical comparisons between the modified model and seven year olds on the nine measures outlined in Chapter 4 will only be given when there is a reliable difference between the scores (see Chapter 6 for discussion on this matter). In addition, any reliable differences between the modified model and the original model will be given, because these indicate the effect of the modification (i.e. they indicate which measures have changed and therefore which measures the modification has influenced).

Theories of development highlight three main areas in which development may occur: knowledge, capacity, and speed. Each of these is examined in turn. A summary of the modifications is then given.

7.2.1 Modifications to knowledge

Most theories (Piaget, Klahr & Wallace, Vygotsky, Pascual-Leone, Case, Halford) present a view that knowledge is consistently built upon, such that new knowledge arises, in part, from existing knowledge. For example, applying the same knowledge to the same problem means an eventual reduction in the time taken to solve the problem (if repeated long enough, a practice effect can be seen, Newell & Rosenbloom, 1981). This is explained by the successive building of new knowledge structures based on the existing knowledge that was used to solve the task (computational models of practice also verify this, such as Nerb, Krems & Ritter [1993]).

Siegler presents a slightly different view. He does not explain how new knowledge arises (merely noting that it does), but does explain in detail how the knowledge is used (which the other theories, in general, fail to do). For example, Siegler proposes that the same problem may be solved on separate occasions using different knowledge, and that children adapt their knowledge to the type of problem they are facing. Whilst these are not readily explained by other theories of development, they are still aspects of development to be explored. Modifications to knowledge need to capture all of these ways in which the theories propose knowledge develops.

Included in knowledge is the representation of the task, the general knowledge acquired that can be applied to the task, and domain-specific knowledge regarding not only facts, skills, and strategies, but also when to apply strategies in given situations. The three aspects of knowledge are covered to different extents in the knowledge based theories of development. Alterations of each aspect will be covered. Modifications that are applicable to one specific theory of development are described first, followed by knowledge modifications that are equally applicable to any knowledge motivated theory of development.

7.2.1.1 Piaget's internal operations

The behaviour of children in the pre-operational and concrete operational stages is important when examining behavioural differences between seven year old children and adults within Piaget's theory. Toward the end of the pre-operational stage (around the age of seven), children can perform operations on internal representations rather than having to carry them out in the external world. All knowledge based theories of development support this view, because new knowledge arises from previous knowledge. However, it is only Piaget who states that this specific type of knowledge appears at around the age of seven.

The model carries out two forms of internal operation which can also be carried out externally: perception of whether an intended fit will result in the construction being flush on its outer edges, and perception of whether any block features will obstruct the fitting of two blocks. Both of these operations are carried out in the model *before* the blocks are physically fit together. Removing these abilities in the model enables the examination of the extent to which the lack of internalisation affects task behaviour.

Removing the check for flush outer edges of an intended fit

Removing the check that an intended construction is flush on its outer edges should mean that more errors are produced than in the original model. The type of error should change because incorrect constructions no longer need to have flush edges. An increase in errors should mean that the task takes longer to complete than it did in the original model.

Table 7.1 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model is within 20% of the seven year olds for construction attempts but not for time taken.

Table 7.1: Construction attempts and time taken for the original model, the same model with internal fitting removed, and seven year old children. Standard deviations are shown in parentheses.

	Original model	No internal fitting model	Seven year old children
Construction attempts	23.1 (2.4)	25.6 (8.8)	27.6 (3.5)
Time taken	129.0 s (31.5)	132.4 s (47.2)	174.0 s (32.1)

The number of task appropriate features for incorrect constructions (i.e. how many features are fit together correctly even though the construction is incorrect, see Chapter 4) drops significantly in comparison to the original model ($t(83)=2.25$, $p<0.05$), because removing the internal check for flush outer edges means that jagged-edged constructions can be made. Jagged-edged constructions are immediately detected as being incorrect, and so the average time taken to disassemble constructions also drops significantly ($t(83)=3.55$, $p<0.01$).

The modification has changed the distribution of task appropriate features for incorrect constructions. Figure 7b shows the distribution of task appropriate features for the original model, the modified model, and the seven year old children. Although the distribution of types of error made by the model is now broader, the frequencies do not match those of seven year old children.

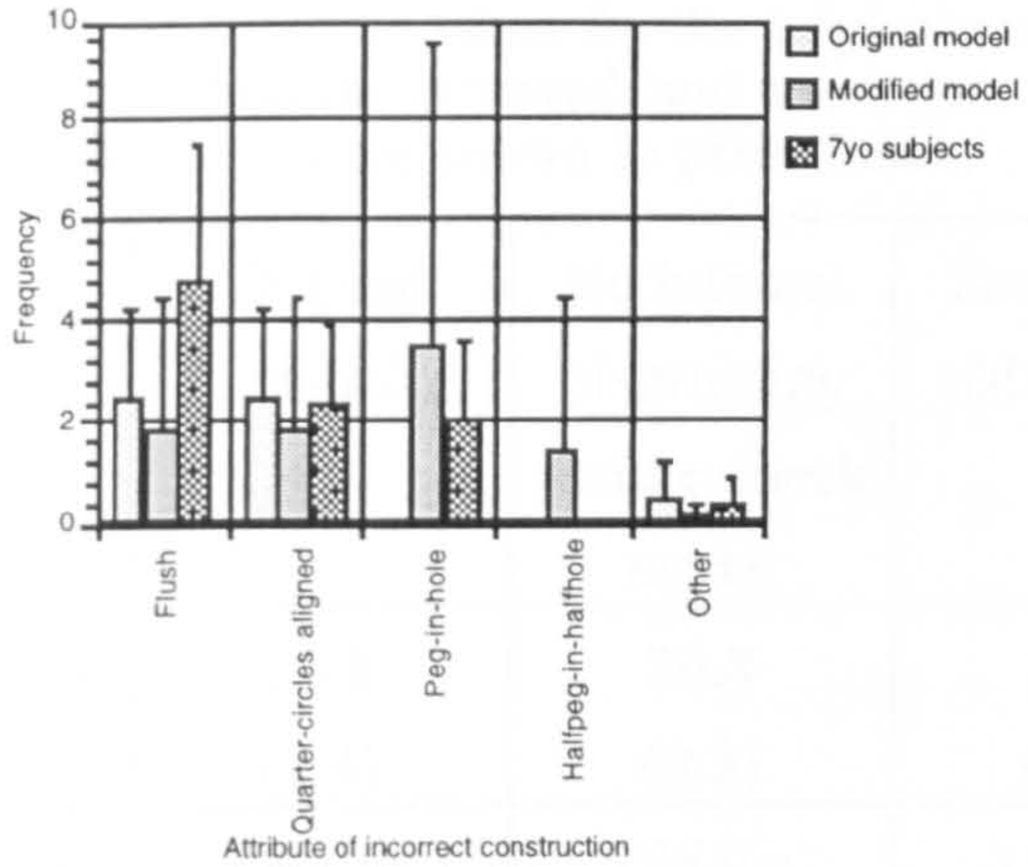


Figure 7b: Frequency distributions for attributes of incorrect constructions.

The removal of the check for flush edges of an intended fit increased scores on those measures that were predicted to increase. The increase does not enable the model to quantitatively match the behaviour of seven year old's on all the predicted measures, as can be seen in Table 7.1. This modification is not enough to enable the model to behave in a similar way to seven year old children.

Removing the check for features obstructing the intended fit

The internal check for any features that obstruct an intended fit should not significantly affect the model's behaviour. This is because the check is only considered if the intended fit is already known to be flush on its outer edges. Even then, there is only a 50% chance of checking for obstructing features (the other option being not to bother checking this). The expectation is therefore that behaviour does not change significantly.

Table 7.2 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modification has significantly reduced both the number of construction attempts ($t(18)=2.85$, $p<0.05$) and the total time taken ($t(18)=2.12$, $p<0.05$) when compared to the original model. There is also a significant difference between both measures when comparing the modified model to the seven year old children.

Table 7.2: Construction attempts and time taken for the original model, the same model with internal obstructing features checking removed, and seven year old children. Standard deviations are shown in parentheses.

	Original model	No internal obstructing feature check model	Seven year old children
Construction attempts	23.1 (2.4)	20.8 (0.8)	27.6 (3.5)
Time taken	129.0 s (31.5)	106.9 s (9.8)	174.0 s (32.1)

The reduction in time taken can be attributed to no longer checking for obstructing features. The implication is that adults often carry out this check even though statistically it will rarely be required. However, the number of construction attempts was expected be the same as for the original model (or increase). The reduction in errors is therefore difficult to comprehend.

Piaget summary

It would be expected that, given this is a physical problem solving task, that internalisation would significantly affect behaviour. However, this is not the case. Removing either of the two internalisation processes in the model has not enabled the model to match the behaviour of seven year old children.

7.2.1.2 Siegler's strategy choice and strategy efficiency

Siegler's theory is the most suited to the Tower task because it focuses on problem solving. The central theme of his theory is adaptive strategy choice: through age and experience, children learn to select the most appropriate strategy for the current problem. If strategy choice is a mechanism of cognitive change, then adults should be more able to select better strategies on the Tower task than seven year old children.

Siegler also believes strategy efficiency changes with age. Older children are able to perform strategies with more accuracy than younger children. Strategy choice and strategy efficiency will be examined by making appropriate modifications to the model.

Strategy choice

The model implements strategies as rules, or sequences of rules (for example, the rules that decide how to fit blocks determine which strategy is used in building a layer).

Selection of a rule uses the expected gain value associated with each rule. The model of adult behaviour has a noise of 0.04 placed on the expected gain value. This means that on some occasions, rules which do not have the highest expected gain value will be selected (when deciding how to fit blocks, this may result in an inappropriate block fitting strategy being selected). Increasing the value of the expected gain noise means that the model has less knowledge about which rules are the most effective. This will enable the examination of the model's behaviour when strategy choice is not as effective as that of the model of adult behaviour. The utility of using expected gain is that it applies to every rule in the model, thereby influencing strategy choice for all of the various strategies involved in constructing the Tower.

Increasing the expected gain noise should mean that more errors occur because it is now more likely that an incorrect strategy will be selected. The time taken to complete the task should also increase, for two reasons: first, the increase in errors should mean more task time; second, the increase in incorrect strategies will mean less intended constructions being flush on their outer edges, which will mean more aborted constructions, and hence more time taken. Strategy use should be more variable, because the noise means that the most appropriate strategies will not always have the highest expected gain value. There would seem to be no reason why the type of incorrect construction should alter, because the internal checks on constructions are still present, which limit the type of incorrect construction that can be produced.

The expected gain noise was increased in steps of one unit, starting at a noise level of one. Figure 7c shows the total time taken at each level of expected gain noise. When the expected gain noise rose above six, the model fails to complete the task on some of the ten runs. This data is therefore not reported.

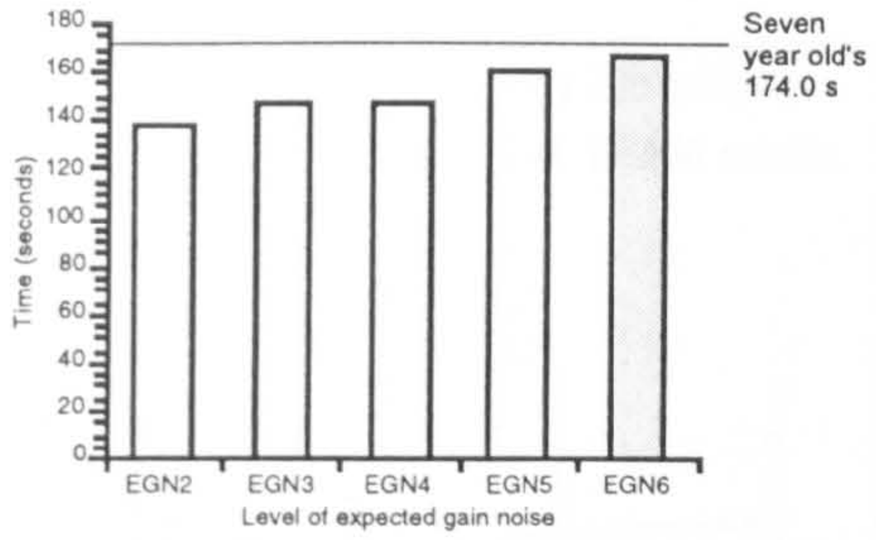


Figure 7c: Time taken for the model to complete the task, at varying levels of expected gain noise.

The time taken to complete the task increases as the expected gain noise increases. The level which enables times to be the closest to those of the seven year olds is when the expected gain noise is set to 6.0. The behaviour of the model at this setting (henceforth the EGN6 Model) will therefore be examined in more detail. Micro-levels of expected gain noise (e.g. 6.1, 6.2), which should push timings closer to those of seven year olds, are not examined because the aim here is to show that modifications to models can change the model's behaviour to move closer to the behaviour of seven year olds. An exact match on every measure is not required to show that modifying architectures is a useful method to examine development (individual modifications are not expected to provide an exact fit anyway).

Table 7.3 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. There is a reliable difference between the original and EGN6 models for time taken ($t(18)=2.39$, $p<0.05$). The EGN6 model is within 20% of the seven year olds for both construction attempts and for time taken, and so layer-by-layer measures will be examined. Figures 7d and 7e show the timings and construction attempts on a layer-by-layer basis. The error bars indicate seven year olds to the left, and the modified model to the right.

Table 7.3: Construction attempts and time taken for the original model, the EGN6 model, and seven year old children. Standard deviations are shown in parentheses.

	Original model	EGN6 model	Seven year old children
Construction attempts	23.1 (2.4)	25.1 (2.9)	27.6 (3.5)
Time taken	129.0 s (31.5)	166.3 s (37.9)	174.0 s (32.1)

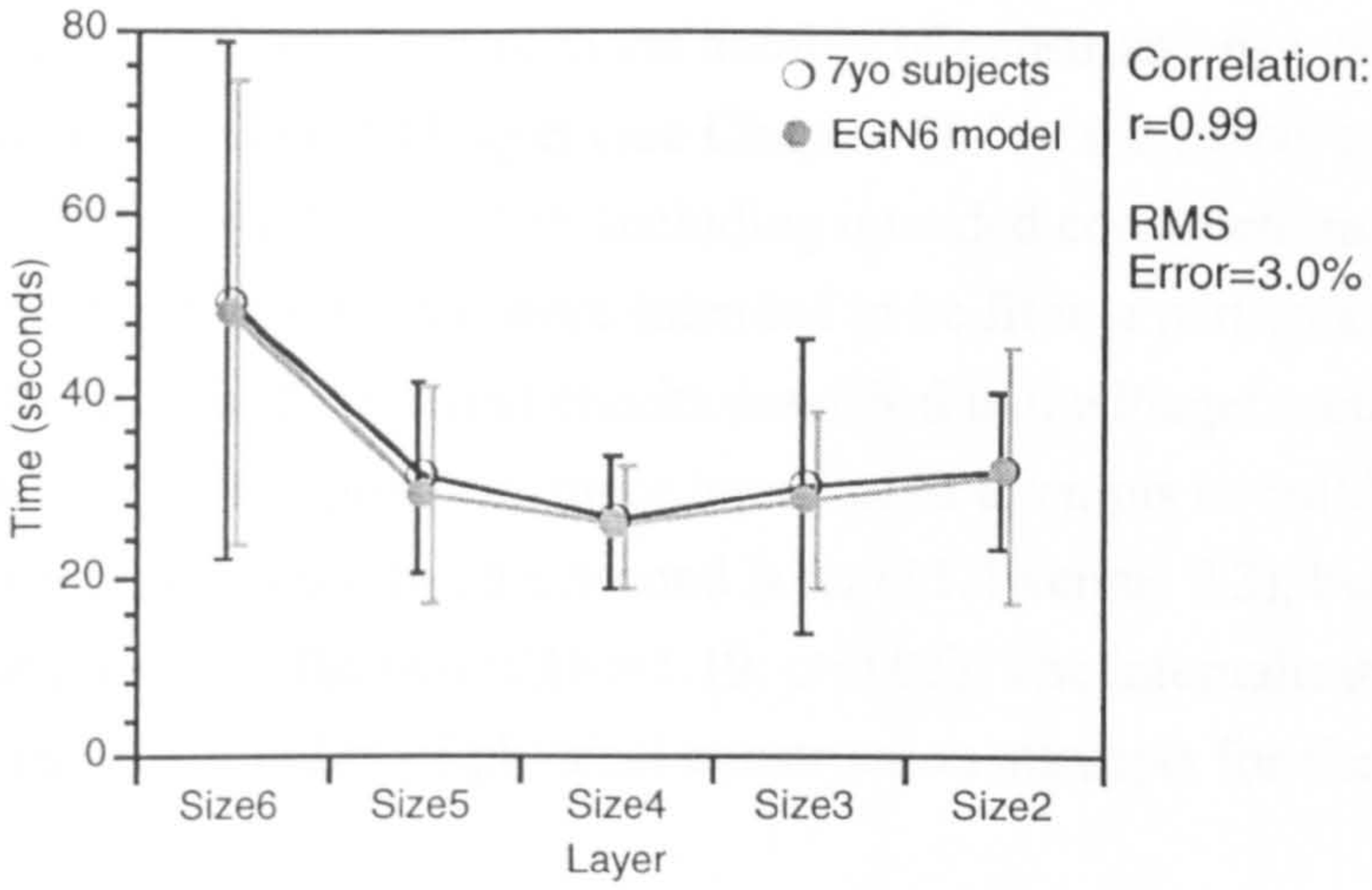


Figure 7d: Time taken to complete each of the five layers, for the EGN6 model, and the seven year old subjects.

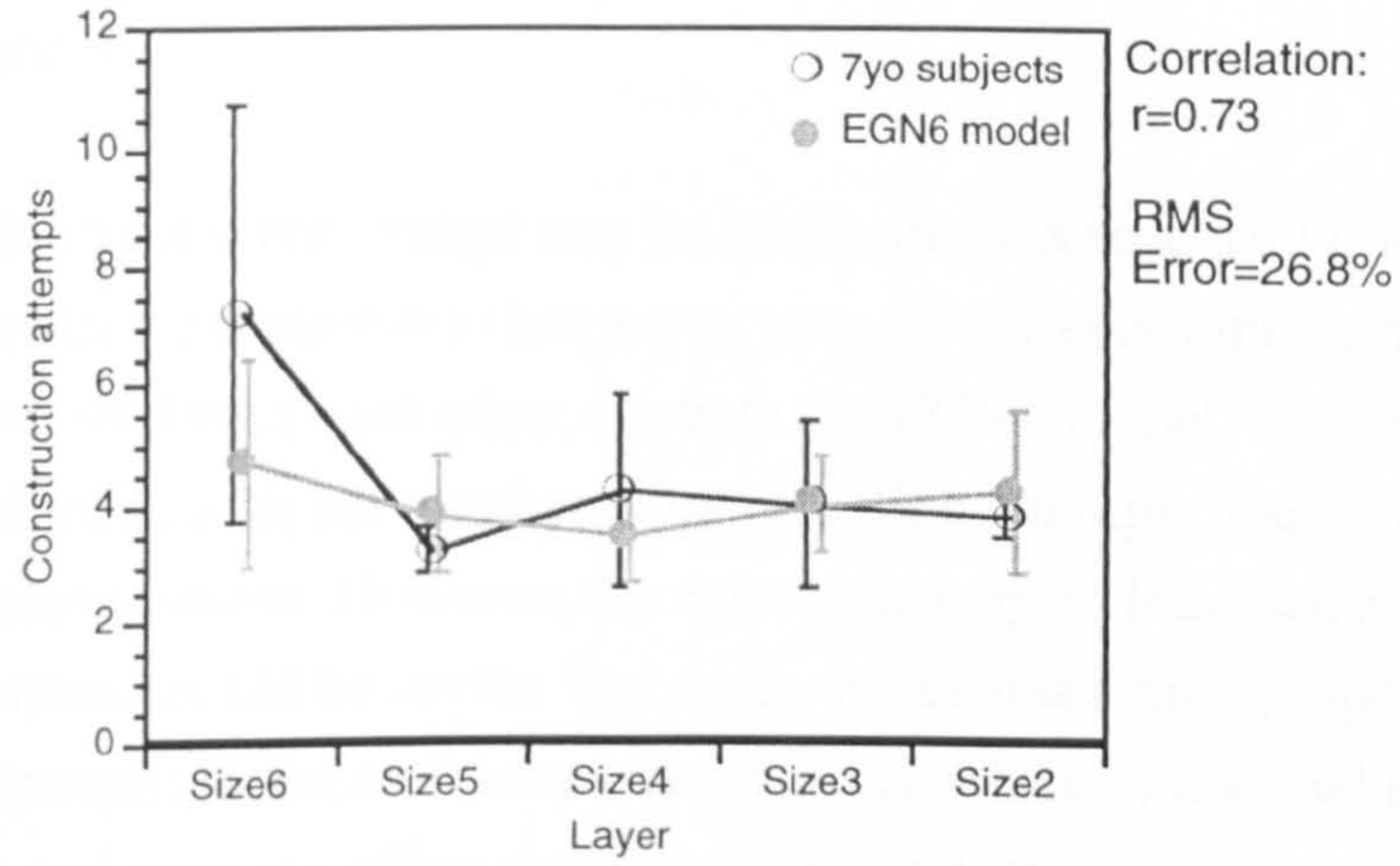


Figure 7e: Constructions made in completing each of the five layers, for the EGN6 model, and the seven year old subjects.

The fit between the EGN6 model's layer times and those of seven year old children's layer timings is very good. Although the fit on layer construction attempts is not as good, it is still a reasonable correlation.

The timings and construction attempts for the very first layer constructed (the largest layer, size6) are interesting, because the EGN6 model and seven year olds match well for time taken but not for number of construction attempts. It should be expected that both sets of measures should match closely, because layer timings and layer construction attempts correlate well for the EGN6 model (r=0.89) and for the seven year old subjects (r=0.91).

There is a significant difference between the number of constructions made by seven year old's for the first layer and second layer (see Chapter 4). For the EGN6 model, there is no reliable difference ($t(18)=1.44$, $p>0.05$). Including intended constructions in the analyses of the EGN6 model (i.e. blocks that were intended to be fit in a particular way, but the fit was abandoned because of the internal checks described in the Piaget section above) shows that the EGN6 model produces more *intended* fit attempts in building the first layer of the Tower than in building the second layer (11.4 versus 9.2), but there is no reliable difference between the two ($t(18)=1.19$, $p>0.05$). The internalisation checks do not account for the low number of physical construction attempts for the first layer.

The earlier prediction that the total number of abandoned (or intended) constructions would rise is borne out, but the increase is only by a small amount. The total number of intended fit attempts is 45.5 (s.d. 7.4) for the original model which rises to 52.3 (s.d. 8.1) for the EGN6 model.

The high correlation for layer timings may be misleading because the EGN6 model makes less physical construction attempts than seven year olds. Layer timings and construction attempts correlate well with each other, for both the EGN6 model, and seven year old's. If the EGN6 model made a higher number of construction attempts, the time to complete the task should also increase. However, the extra time may be less than expected (because the increase in errors should be for the first layer, which has a high proportion of abandoned fit attempts; if some of these were to become errors, it would require minimal extra task time), and may not offset the correlation anyway.

The modification has changed the layer strategy distribution, although the original model still provides a closer match to the seven year olds than does the EGN6 model (see Figure 7f). The only reliable difference is between the original and EGN6 models, for the peg/hole strategy ($t(18)=2.61$, $p<0.05$).

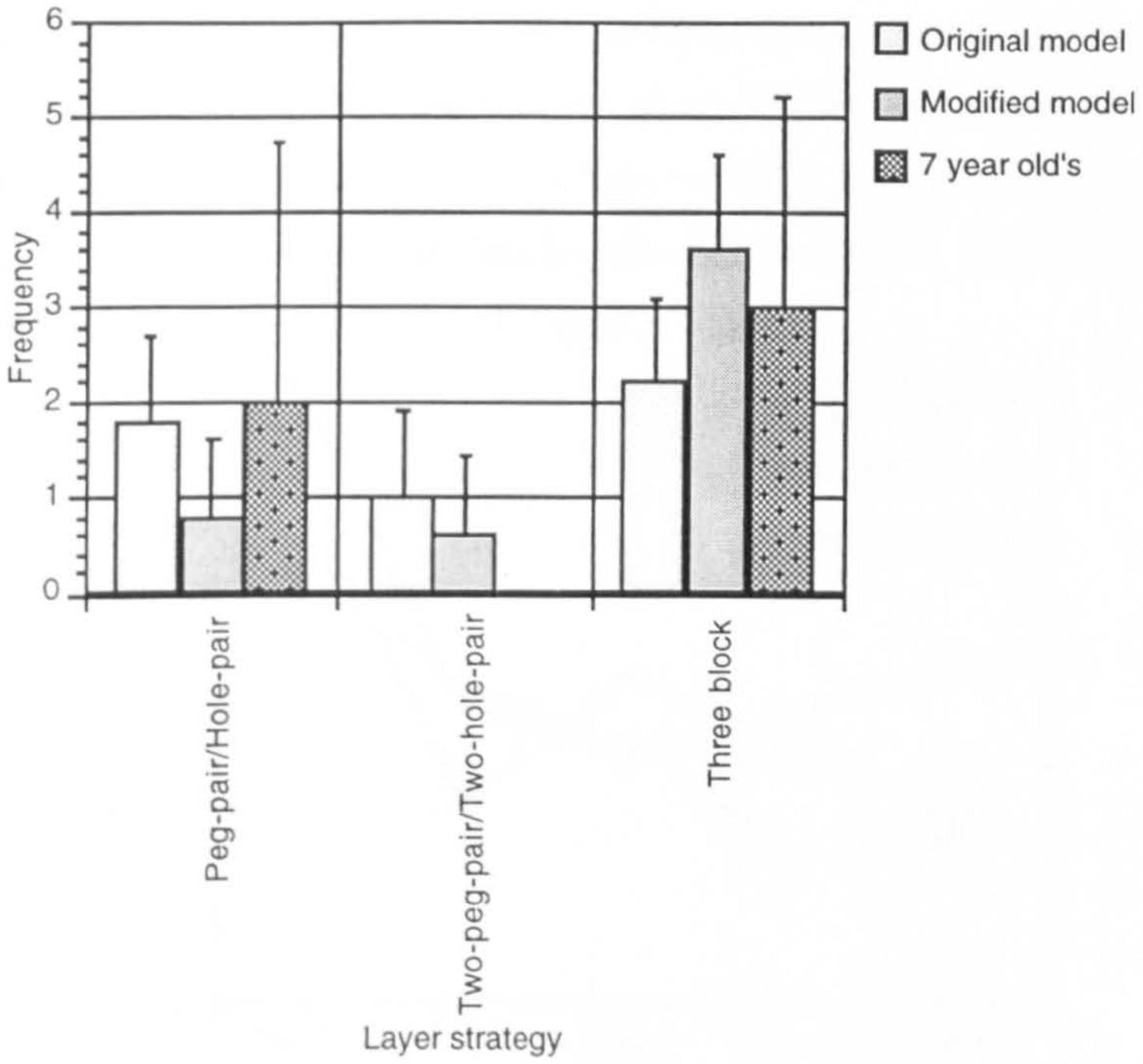


Figure 7f: Frequency of layer strategies for the original model, the EGN6 model, and seven year old children.

The EGN6 model provides the closest fit to the seven year old children for time taken to complete the Tower. At higher levels of expected gain noise, the model did not complete the task. Expected gain noise cannot be the sole mechanism of development for two reasons. First, the EGN6 model does not fully match the seven year old behaviour. Second, in order for the model to subsequently match the performance of, say, five year old children, the noise would have to be increased. At higher levels the model cannot complete the task (although most five year olds require instruction to complete the task, see Chapter 4).

Expected gain noise must interact with other modifications to enable the model's behaviour to fit the behaviour of seven year olds. To model the behaviour of seven year old children, the interactions should use low levels of expected gain noise (e.g. level three), because higher levels (such as six) will represent children younger than seven years old. The lower levels of expected gain noise (EGN1-EGN5) must be examined here to see whether they show the same pattern of results as the EGN6 model. This will mean that they can usefully be used when interacting the modifications in Chapter 8.

Figures 7g and 7h show models EGN2, EGN4, and EGN6 (rather than show all six models, a suitable range is selected for clarity). For layer timings, every model correlates

well with the layer timings for seven year old children (average r=0.80, range 0.61–0.99). For layer construction attempts, the models are more haphazard, ranging from no correlation at all, to a correlation of 0.93 (average r=0.50). Using lower levels of expected gain noise will probably show correlations for the time taken to produce each layer, but not for the construction attempts for each layer.

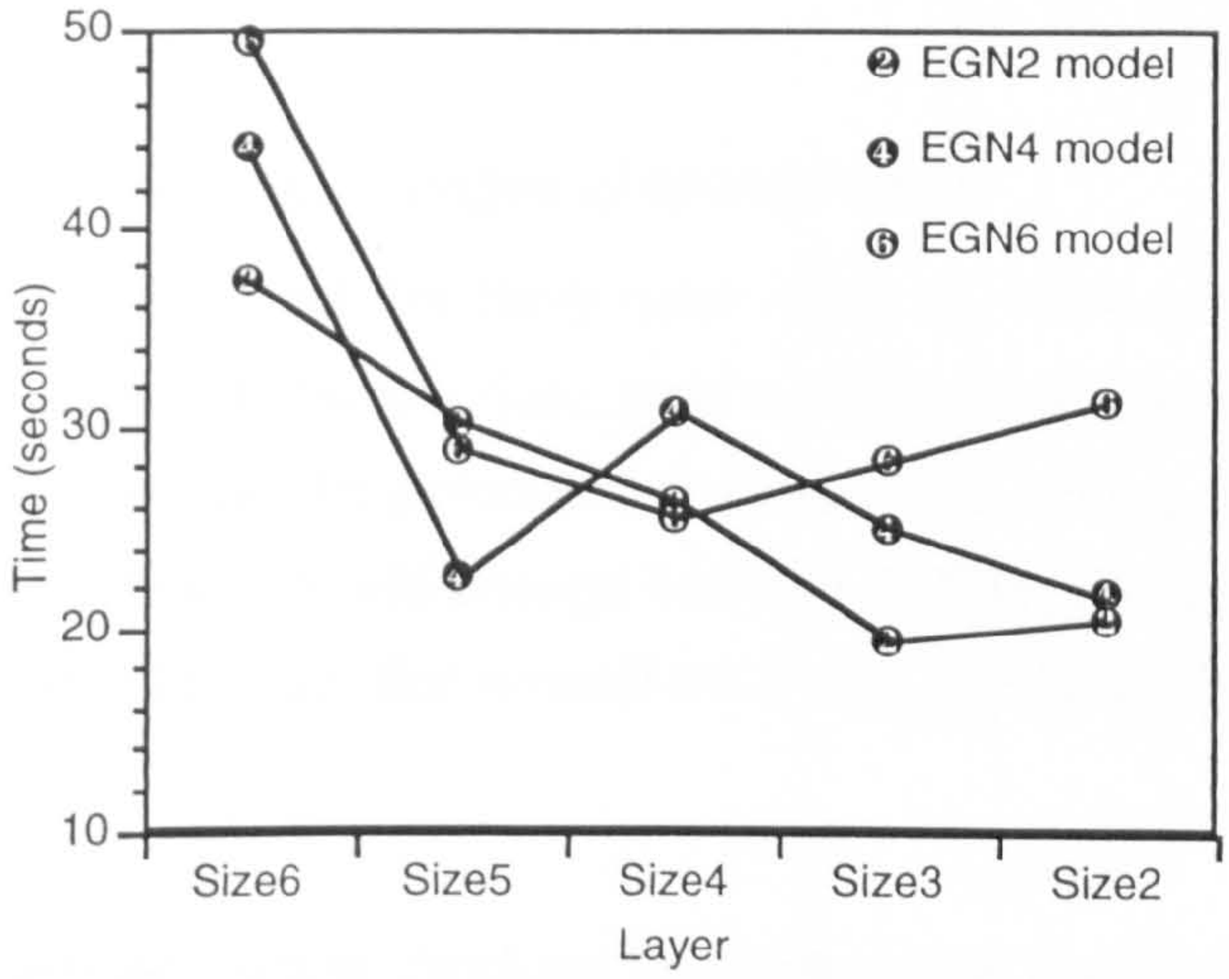


Figure 7g: Time taken to complete each of the five layers, for the EGN2, EGN4, and EGN6 models.

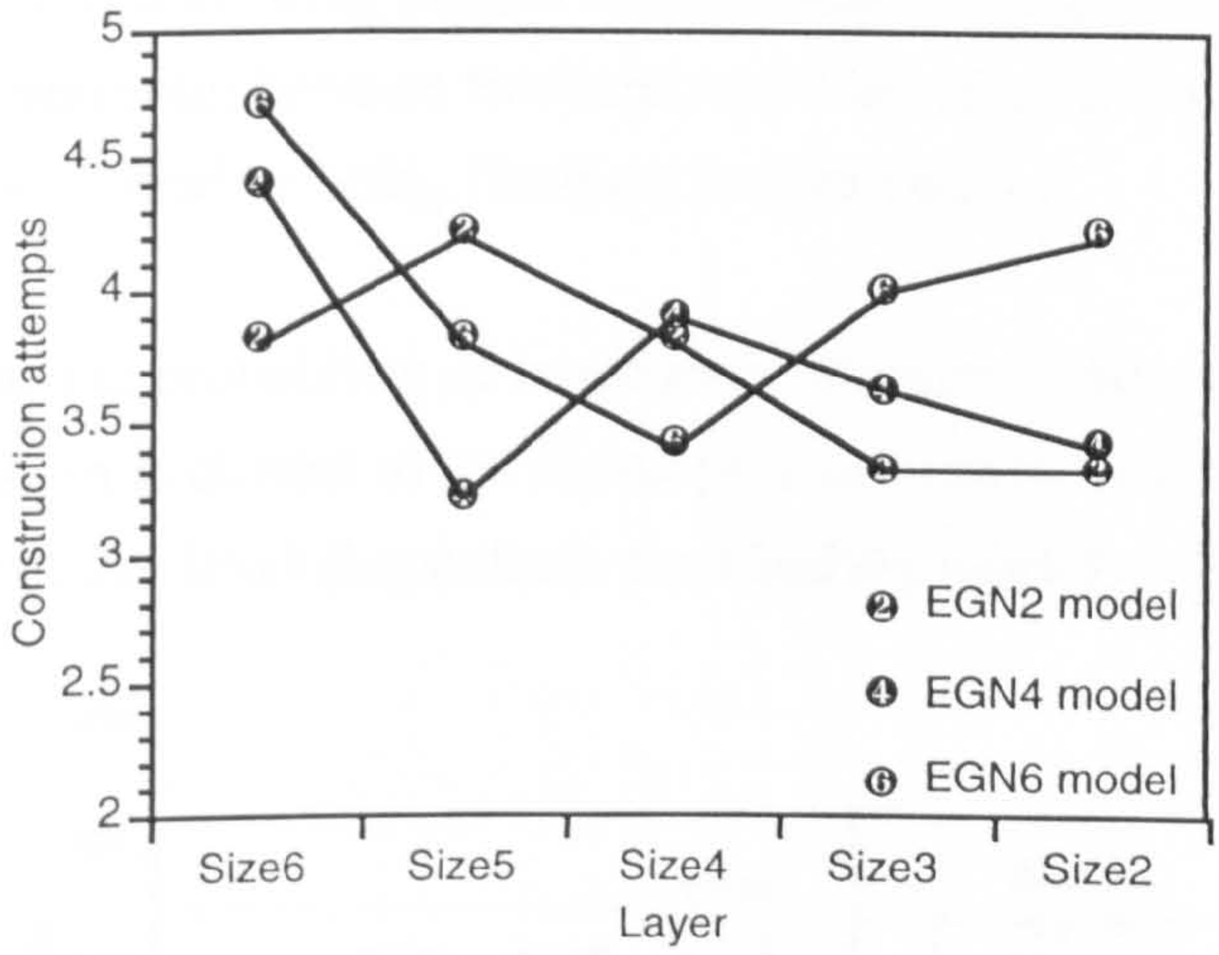


Figure 7h: Constructions made in completing each of the five layers, for the EGN2, EGN4, and EGN6 models.

Examining the different levels of expected gain noise shows that lower values of EGN can be used profitably when interacting modifications. The strategy efficiency part of Siegler's theory will now be examined.

Strategy efficiency

The model has two parameters whereby accuracy can be altered for two different strategies: the check that an intended fit has flush outer edges, and the check for features obstructing the intended fit. These strategies occur before fitting two blocks, the first to ensure that an intended fit is flush, and the second to ensure nothing obstructs fitting the blocks together.

Accuracy of the check for flush outer edges of an intended fit

Reducing the accuracy of the check for flush outer edges of an internal fit will mean that there should be more incorrect constructions produced (because intended constructions that are not flush will sometimes be perceived as being flush, thus passing the check). The type of incorrect constructions should change because of this. The proposed increase in construction attempts should mean that overall time also increases, because the two are correlated.

The probability of inaccuracy when checking for flush edges was varied from 10% to 50%, where 10% represents a probability of one in ten that the construction will be perceived inaccurately (i.e. the intended fit is perceived as being flush when it is not, or the intended fit is perceived as being jagged when in fact it is flush). An inaccuracy level of 50% is a realistic maximum because this represents the chance level that a random construction will be perceived as being flush on its outer edges.

As Figure 7i shows, as the probability of inaccuracy increases, the overall time taken increases. The time taken is closest to the seven year old subjects at the 40% inaccuracy level. This behaviour at this level (henceforth the Flush40 model) will now be expanded.

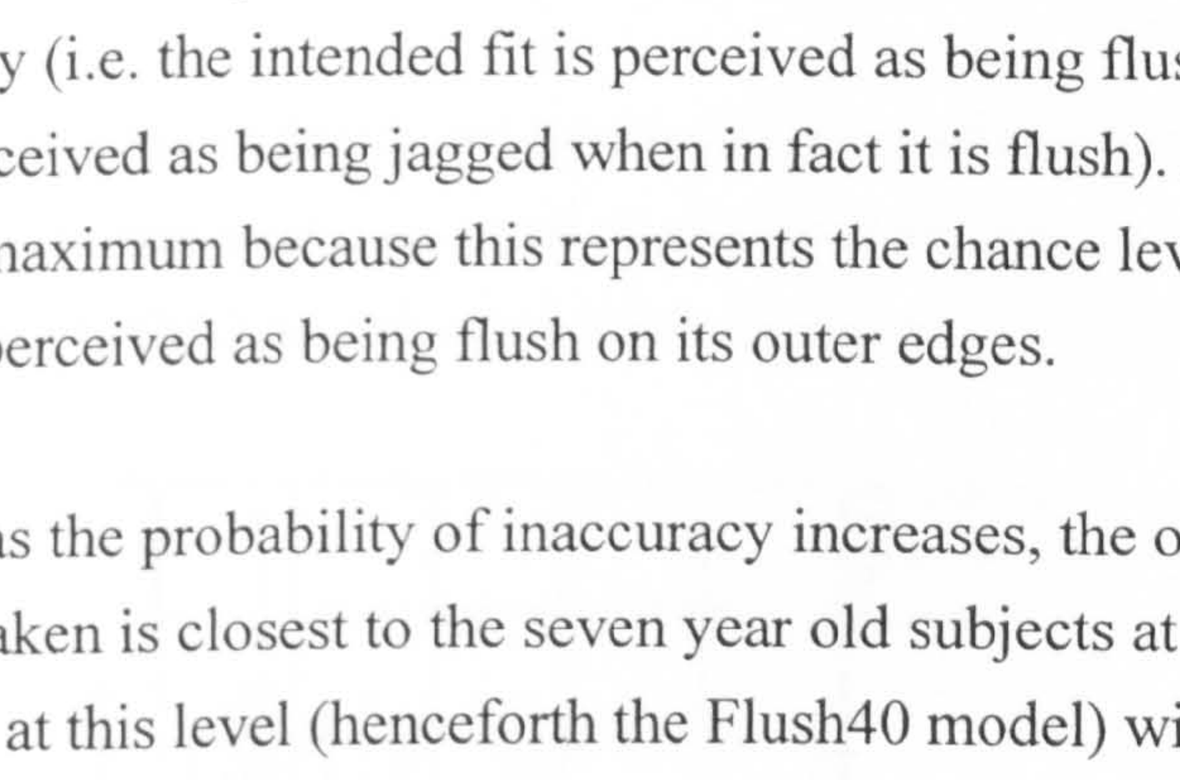


Figure 7i: Time taken for the model to complete the task, at varying levels of accuracy of the flush fitting check.

Table 7.4 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. Both overall time and the number of construction attempts significantly increase for the Flush40 model over the original model (respectively, $t(18)=2.48$, $p<0.05$; $t(18)=3.59$, $p<0.01$). The Flush40 model is within 20% of the seven year olds for both construction attempts and time taken. The Flush40 model correlates with the seven year old children reasonably well for the time taken to construct each layer ($r=0.72$) but does not correlate for the number of construction attempts each layer ($r=0.19$).

Table 7.4: Construction attempts and time taken for the original model, the Flush40 model, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Flush40 model	Seven year old children
Construction attempts	23.1 (2.4)	26.1 (3.0)	27.6 (3.5)
Time taken	129.0 s (31.5)	183.3 s (35.9)	174.0 s (32.1)

Figure 7j shows that the type of incorrect construction (denoted by the task relevant features involved in each incorrect construction) differs between the original and Flush40 models, because in the Flush40 model, not all incorrect constructions have flush outer edges.

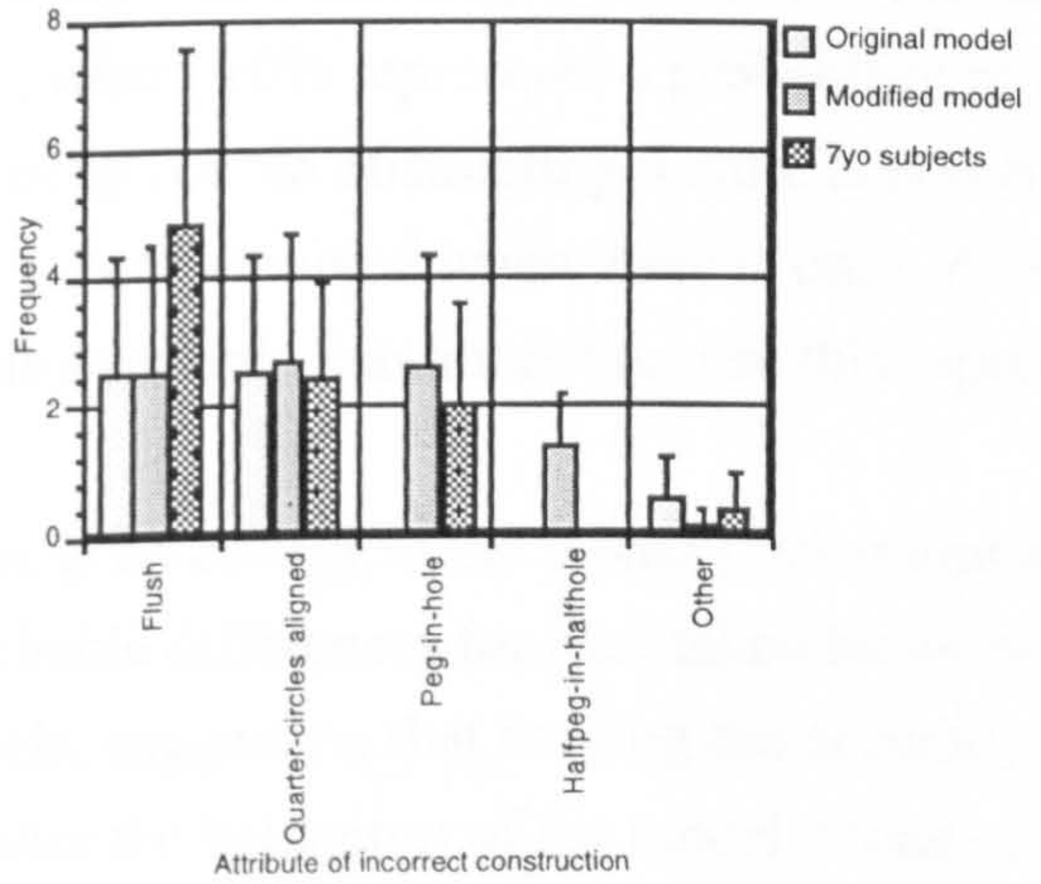


Figure 7j: Frequency distributions for attributes of incorrect constructions.

Altering the probability of inaccuracy when fitting blocks together (i.e. altering the accuracy of internal fitting) is not the same as removing the internal fitting process altogether (as for the Piaget modification). Removing the process should lead to nearly every intended construction being physically attempted. Altering the accuracy means that

some intended fit attempts which are not flush will still be attempted, and also some which are flush will be aborted. Comparing the two shows that both attempt a similar number of constructions (25.6 for the no-internal-fitting model versus 26.1 for the Flush40 model), yet the time taken is vastly different (132.4 s versus 183.3 s respectively). There is a reliable difference for time taken ($t(18)=2.71$, $p<0.05$). The additional time taken for the Flush40 model must, in part, come from the aborted fit attempts which would actually result in flush outer edges.

The Flush40 model shows that accuracy of strategies may well be a factor in development, because it enables both errors and time taken to increase to a level which matches the seven year old children. The type of incorrect construction is closer to the seven year old children than the original model, although the model produces too many incorrect constructions using halfpegs and halfholes, which seven year olds never do.

Accuracy of the check for features obstructing the intended fit

Altering the accuracy of the check for features obstructing the intended fit should not significantly alter the behaviour of the model, because the check is rarely carried out. The number of construction attempts should increase slightly, although the distribution of the type of incorrect construction will not change because all incorrect constructions will still have flush outer edges.

The probability of inaccuracy when checking for features obstructing the intended fit was varied from 10% to 50%, where 10% represents a probability of one in ten that the check will be carried out incorrectly (i.e. an obstructing feature is perceived when there isn't one, or an obstructing feature is not perceived when there is one). As with the flush check, an inaccuracy level of 50% is a realistic maximum because this represents the chance level.

As Figure 7k shows, there is no change in the model's behaviour at different levels of accuracy. There are no reliable differences for time taken between the original model and any of the accuracy models, suggesting that varying the accuracy of the obstructing features check does not alter the behaviour of the model. None of the models are examined in further detail.

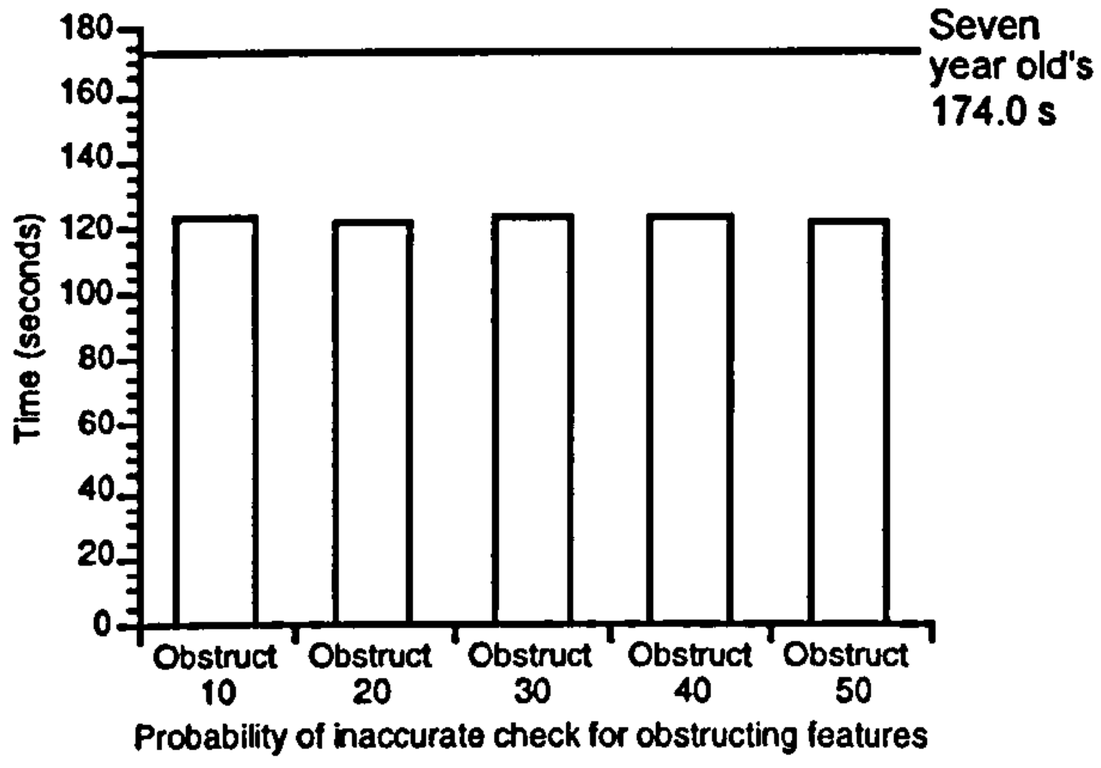


Figure 7k: Time taken for the model to complete the task, at varying levels of accuracy of the obstructing features check.

Siegler summary

Strategy choice and strategy efficiency changes have led to models that are able to approximate the behaviour of seven year old children. Varying the levels of choice and efficiency led to different behaviour in the models, with only one level of strategy choice (EGN6), and one level of strategy efficiency (Flush40), suitably matching the seven year old data. However, models which used different levels were seen to show similar results but of a different magnitude.

7.2.1.3 All theories

All theories of development state that the knowledge base increases with age, and that new knowledge builds upon that which has been previously acquired (although in Halford's specification, declarative knowledge does not necessarily draw upon any previous knowledge). There are two types of knowledge in the model: rules and facts. Modifications to this knowledge (i.e. removal of part of it) must ensure that the rules or facts are complex enough to have been constructed from other knowledge.

Procedural knowledge

In deciding which procedural knowledge to remove, the five behaviour modules (described in Chapter 5) should be borne in mind. Knowledge changes should cover as many modules as possible so that the type of knowledge that is modified is spread across the whole model. The Fit Together module has already been changed in examining aspects of the Piagetian and Siegler theories, so the remaining four modules are considered for knowledge changes.

The model uses two subtle strategies in searching for and fitting blocks. When searching, each block that is examined is remembered so that it need not be examined again. When

two blocks are fit together, and the fit failed, the blocks and features involved are remembered so that they will not be tried again.

There are also different levels of search criteria that can be used (size, or size together with specific features). Searching by size together with features is better than searching by size only, because it increases the chances of obtaining a pair of blocks which can be fit together successfully.

Removing each of these three types of procedural knowledge (remembering blocks seen, remembering previous fit attempts, and searching by features) will be examined. These three changes cover the Scan Table, Decide Search, and Decide Fit modules. The Assess Fit module decides whether or not a construction is correct. Children of three years and upwards can determine a correct construction from an incorrect one (Wood, Bruner & Ross, 1976), and so the Assess Fit module is not considered for knowledge changes.

Removing the strategy for remembering blocks seen

Keeping track of which blocks have been examined is important for visual scanning of the blocks. Removing this ability should mean that scanning takes longer, and involves more fixations. This should increase the task time, although the number of construction attempts should not alter.

Table 7.5 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model differs significantly from the seven year old's for both construction attempts ($t(13)=4.74$, $p<0.01$) and time taken ($t(13)=4.68$, $p<0.01$).

Table 7.5: Construction attempts and time taken for the original model, the model which does not remember blocks seen, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Modified model	Seven year old children
Construction attempts	23.1 (2.4)	21.8 (1.3)	27.6 (3.5)
Time taken	129.0 s (31.5)	114.1 s (18.2)	174.0 s (32.1)

The removal of the strategy for remembering blocks seen has not altered the total number of construction attempts or the total time taken. The probable reason for this is that the model gathers all the blocks of the same size that it can see. Often the blocks that have just been gathered are selected for fitting. This reduces the amount of visual search required. For each layer constructed by the original model and the modified model, there is no appreciable difference in the number of fixations that are required (average of 22.8 and 21.0 respectively). There is also no appreciable difference between the original and modified models in the time spent looking for the blocks of each layer (average of 5.4 and 4.9 respectively). The strategy of gathering blocks of the same size, which both adults and seven year old children use, appears to be one which reduces the need for remembering blocks that have been seen.

Removing the strategy for remembering previous fit attempts

In order that an attempted construction is not repeated, the blocks and features involved in the fit are recorded in memory. Neglecting to record these details should mean that errors are increased because the same erroneous constructions can be repeated. The increase in errors should mean time increases.

Table 7.6 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model differs significantly from the seven year old's for both construction attempts ($t(13)=6.47$, $p<0.01$) and time taken ($t(13)=5.10$, $p<0.01$).

Table 7.6: Construction attempts and time taken for the original model, the model which does not remember fit attempts, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Modified model	Seven year old children
Construction attempts	23.1 (2.4)	20.4 (0.7)	27.6 (3.5)
Time taken	129.0 s (31.5)	115.6 s (13.2)	174.0 s (32.1)

The same pair of blocks are never involved in an incorrect construction more than once, and therefore the same error never occurs for the same pair of blocks. This is likely to be because so few errors are produced by the modified model. The removal of the strategy

for remembering fit attempts has actually reduced the total number of construction attempts and the total time taken.

Removing search by features

Restricting search to size only should mean that there is more opportunity for error, because blocks with random features will be selected rather than those fulfilling some feature criteria. The increase in construction attempts should lead to an increase in task time.

Table 7.7 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model differs significantly from the seven year old's for both construction attempts ($t(13)=3.45$, $p<0.01$) and time taken ($t(13)=4.92$, $p<0.01$). Searching using size as the only criteria has not changed the total number of construction attempts or the total time taken.

Table 7.7: Construction attempts and time taken for the original model, the model which does not search by features, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Modified model	Seven year old children
Construction attempts	23.1 (2.4)	23.3 (1.4)	27.6 (3.5)
Time taken	129.0 s (31.5)	121.2 s (9.8)	174.0 s (32.1)

Only using size in the search criteria may not influence the model's behaviour because of the gathering strategy mentioned above. Only when the model has failed to gather all four blocks of a size does it fall back on visual search strategies. This will generally mean that searching by size, or searching by size and one feature, are likely to be equally effective in the context of the model's problem solving approach.

Procedural knowledge modifications summary

None of the changes to procedural knowledge significantly altered the behaviour of the model on measures of total number of construction attempts or total time taken. Modifications to declarative knowledge will now be considered.

Declarative knowledge

The model begins with knowledge of features and how they fit together. Children may not have detailed knowledge of the more intricate task features (e.g. halfpegs). The seven year old children, for example, make a sum total of 27 peg-pair or hole-pair constructions, as opposed to a sum total of 8 two-peg-pair or two-hole-pair constructions (the adult subjects make a sum total of 20 of each type). The significance of this is that peg-pairs and hole-pairs require placing the peg of one block into the hole of another to produce the pair, whereas the other types of pairs involve fitting a halfpeg into a halfhole (see Chapter 4).

Three different facts regarding features that fit together will be removed: the fact that halfpegs can align with each other and halfholes can align with each other; the fact that halfpegs fit into halfholes; the fact that quarter circles align with each other.

Removing facts stating halfpegs align and halfholes align

Fitting two blocks of the same size by aligning their halfpegs, or aligning their halfholes, represents a 100% chance of success (see Chapter 4). Removing this declarative knowledge should therefore increase the number of errors, which in turn should increase task time. The distribution of layer strategies should change because less pairs will be constructed by alignment (i.e. there should be less peg-pair and hole-pair constructions).

Table 7.8 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model differs significantly from the seven year old's for both construction attempts ($t(13)=4.69$, $p<0.01$) and time taken ($t(13)=4.45$, $p<0.01$). The removal of the alignment knowledge has not changed the number of construction attempts or the time taken.

Table 7.8: Construction attempts and time taken for the original model, the model excluding feature alignment facts, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Modified model	Seven year old children
Construction attempts	23.1 (2.4)	21.9 (1.3)	27.6 (3.5)
Time taken	129.0 s (31.5)	121.8 s (14.4)	174.0 s (32.1)

The modified model has a different layer strategy distribution than the original model. The number of peg-pair or hole-pair constructions produced has dropped as predicted (from 1.8 in the original model to 0.0 in the modified model; seven year olds average 2.0). However, the modified model does not match the layer strategy distribution of the seven year old children any better than the original model.

Removing facts stating that halfpegs fit into halfholes

Fitting two blocks of the same size by placing the halfpeg of one into the halfhole of the other represents a 50% chance of success (see Chapter 4). This represents chance level and therefore no change in the number of construction attempts should occur. This means that the time taken should also remain constant. However, the layer strategy distribution should change, because less two-peg and two-hole pairs should be produced.

Table 7.9 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model differs significantly from the seven year old's for both construction attempts ($t(13)=4.72$, $p<0.01$) and time taken ($t(13)=4.29$, $p<0.01$). The removal of the halfpeg/halfhole fitting fact has not changed the number of construction attempts or the time taken.

Table 7.9: Construction attempts and time taken for the original model, the model excluding the halfpeg fitting into halfhole fact, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Modified model	Seven year old children
Construction attempts	23.1 (2.4)	22.2 (0.9)	27.6 (3.5)
Time taken	129.0 s (31.5)	126.0 s (12.0)	174.0 s (32.1)

The modified model has a different layer strategy distribution than the original model. The number of two-peg or two-hole constructions produced has dropped as predicted (from 2.4 in the original model to 1.8 in the modified model; seven year olds average 1.6). However, the modified model does not match the layer strategy distribution of the seven year old children any better than the original model.

Removing facts stating that quarter circles align with other quarter circles

Fitting two blocks of the same size by aligning the quarter circles on the blocks represents a 33% chance of success (see Chapter 4). Removing this knowledge should mean that the

number of errors decreases, and therefore the time taken should also decrease. The layer strategy distribution should remain the same, because quarter circle alignment can be applied equally well to any layer strategy.

Table 7.10 shows the total number of construction attempts and the total time taken in constructing the tower, for the models and the seven year olds. The modified model differs significantly from the seven year old's for both construction attempts ($t(13)=4.55$, $p<0.01$) and time taken ($t(13)=4.15$, $p<0.01$). The removal of the quarter circle alignment fact has not reliably changed the number of construction attempts or the time taken.

Table 7.10: Construction attempts and time taken for the original model, the model excluding the quarter circle alignment fact, and seven year old children. Standard deviations are shown in parentheses.

	Original model	Modified model	Seven year old children
Construction attempts	23.1 (2.4)	21.9 (1.4)	27.6 (3.5)
Time taken	129.0 s (31.5)'	119.2 s (19.6)	174.0 s (32.1)

Declarative knowledge modification summary

None of the declarative knowledge changes have significantly altered the behaviour of the model. This may be because the learning mechanism is able to adapt when knowledge is missing. The productions that lead to success become apparent faster because there are less productions competing than before (some productions cannot fire because the relevant knowledge has been removed). The declarative knowledge modifications seemed the most plausible to make, but do not lead to any change in behaviour. With some declarative knowledge and a good learning mechanism, the task can be completed just as well as if the all declarative task knowledge existed in the model. This suggests that seven year old children may have less powerful learning mechanisms than adults.

7.2.2 Modifications to capacity

Three theories suggest capacity is a limiting factor in development. Each theory explains capacity in a different way. To Pascual-Leone, capacity is the number of knowledge structures that can be active at any one time. To Case, capacity remains the same throughout childhood, but through experience it can be used more efficiently (e.g. chunking knowledge means the same knowledge can be represented using less capacity).

To Halford, capacity is in terms of dimensions, where only knowledge structures of a specific complexity (where complexity is limited by number of dimensions) can be represented.

One problem that is noted with capacity-style theories is the definition of what constitutes one item in working memory (Flavell, 1978). Formalising the theory in computational terms forces a decision as to what one working memory item is, because memory items are highly specified in most computational models. A working memory element in ACT-R is a declarative memory element, or DME (see Chapter 5).

Using a DME as the lowest form of working memory element, two immediate capacity limitations are evident in the model. First, the DME's in the model are subjected to decay. The number of DME's that are active at any one time are those that are above a retrieval threshold. Therefore the number of active DME's can be altered by changing the retrieval threshold. Second, limiting the number of conditions (or DME's) that a production rule can have in its condition side may mean that some strategies cannot be used, or have to be implemented as more than one production rule (if possible). Both of these will be explored when examining capacity limitations.

7.2.2.1 Pascual-Leone's M-Power

Cognitive change in Pascual-Leone's theory states that memory capacity, indicated by M-power, increases with age. M-power indicates how many knowledge structures children can hold in memory at any one time (i.e. how many knowledge structures can be active). As M-power increases, more knowledge structures can be active. This will lead to better performance because children can, for example, co-ordinate more task elements and hence make their behaviour more efficient.

The model has a working memory which mediates how many knowledge structures can be active at any one time. All DME's which are above retrieval threshold are considered active, whilst all other DME's are considered inactive. The retrieval threshold parameter can therefore be used to examine the effect of having different amounts of DME's active in the model. The parameter will be used to examine the effect in behaviour of the model at different levels of M-power.

The model's retrieval threshold parameter is varied from two to nine (because ten is the base level of activation for DME's), in steps of one unit. The total time taken, for each level of retrieval threshold, is shown in Figure 7l. The figure only shows threshold values

up to six because the model failed to complete the task when the threshold was set to seven or more. This is because too few elements remain active for sufficient periods of time.

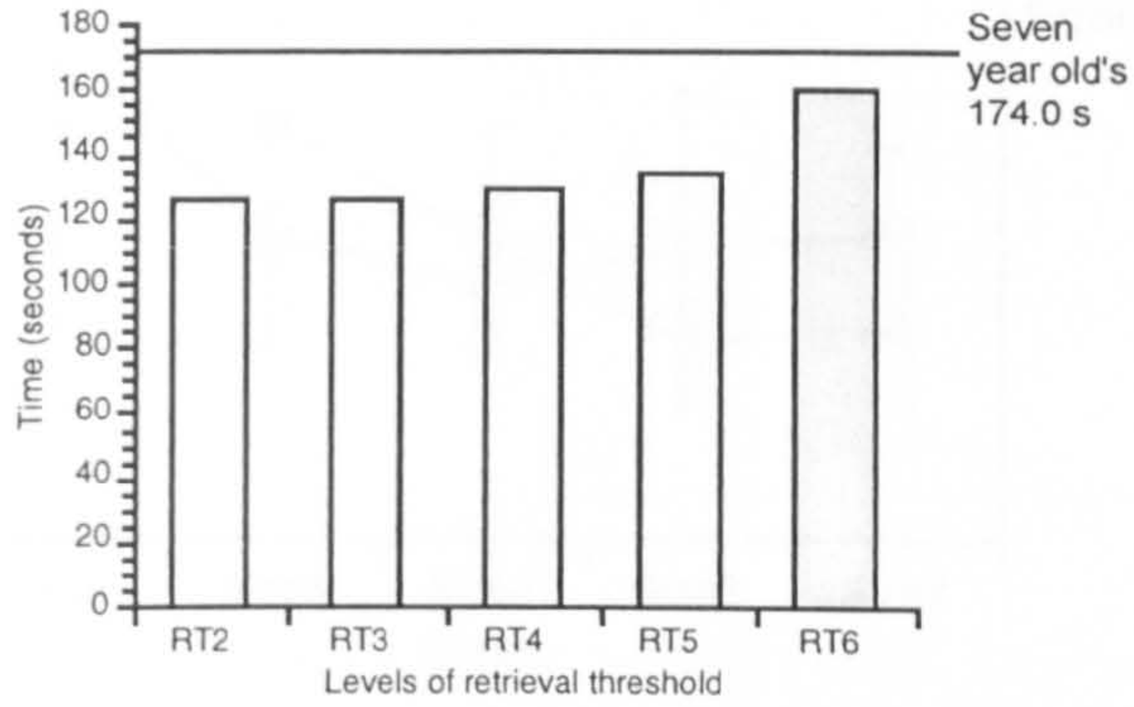


Figure 7l: Time taken for the model to complete the task, at varying levels of retrieval threshold.

The completion time gradually increases, yet never quite manages to reach the time taken by seven year old children. The closest time is when the retrieval threshold is set at six. The behaviour of the model at this setting (henceforth the RT6 Model) will therefore be examined in more detail. Table 7.11 shows the number of construction attempts, and the time taken, in producing the Tower.

Table 7.11: Construction attempts and time taken for the original model, the RT6 model, and seven year old children. Standard deviations are shown in parentheses.

	Original model	RT6 model	Seven year old children
Construction attempts	23.1 (2.4)	25.8 (3.7)	27.6 (3.5)
Time taken	129.0 s (31.5)	160.2 s (35.6)	174.0 s (32.1)

The RT6 model is within 20% of the seven year old children's scores for both time taken and the number of construction attempts. The measures will be examined further by looking at them on a layer-by-layer basis. Figures 7m and 7n show the time taken to produce each layer, and the number of construction attempts taken in producing each layer, for the RT6 model and seven year old children. The RT6 model does not correlate well with the seven year old's on either measure.

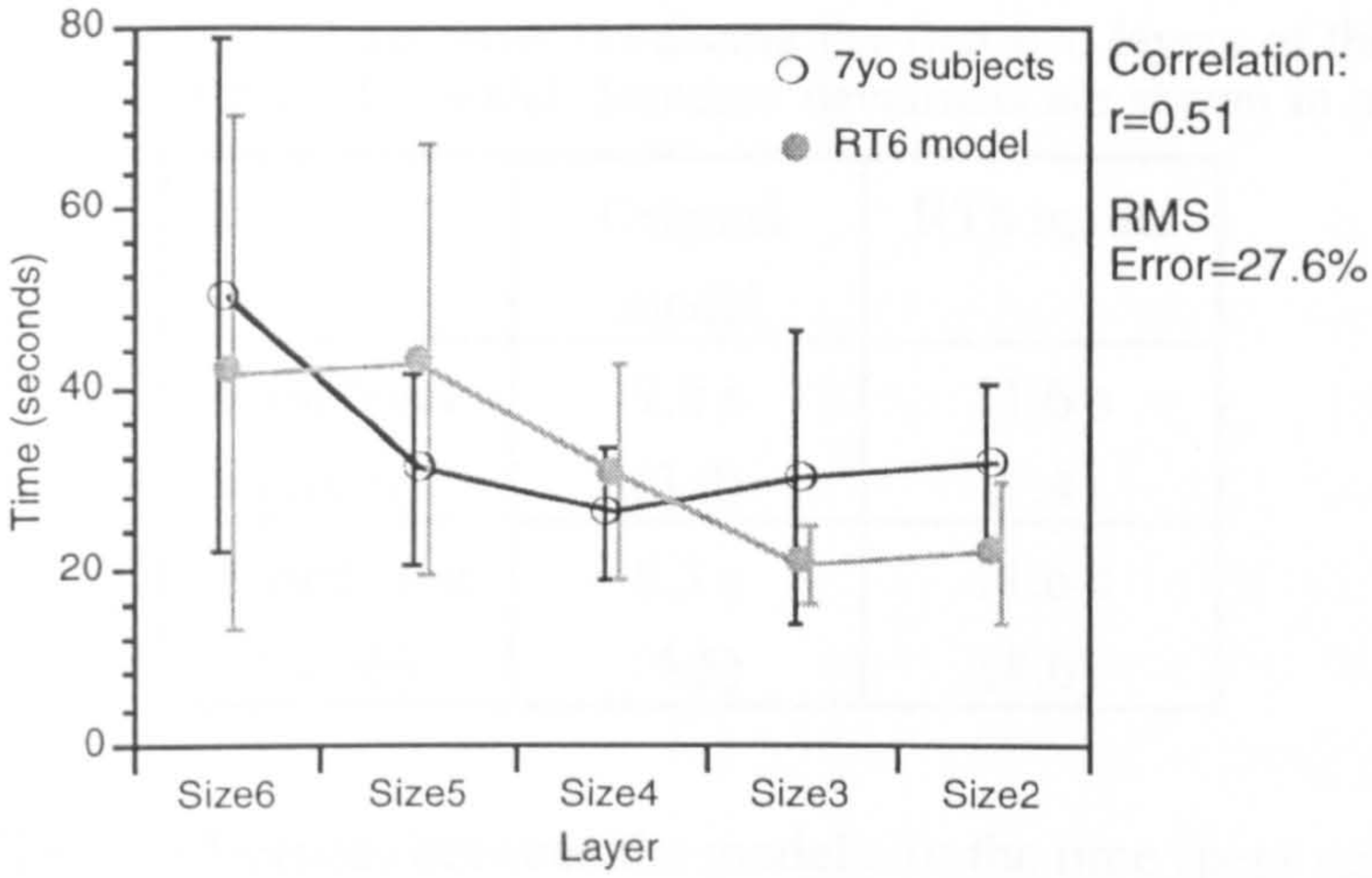


Figure 7m: Time taken to complete each of the five layers, for the RT6 model, and the seven year old subjects.

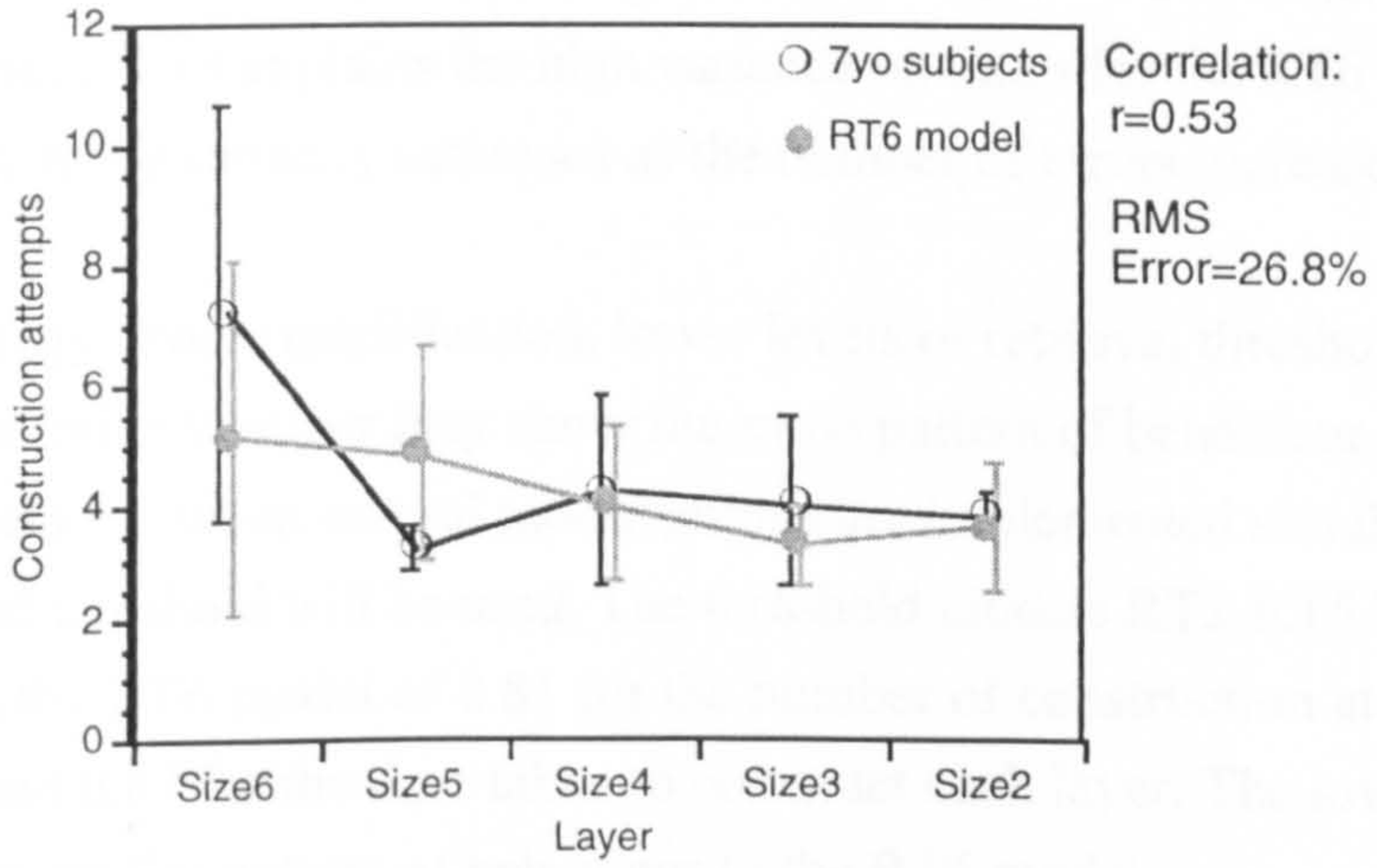


Figure 7n: Constructions made in completing each of the five layers, for the RT6 model, and the seven year old subjects.

The RT6 model takes a similar amount of time in producing the first and second layers of the Tower. This suggests that perhaps a minimum amount of memory is required to manipulate the blocks when there are many blocks on the table. The best way to examine this is to look at the time spent searching the table, because more memory should mean less search time (because block characteristics and locations will be remembered). The search times for the first two layers are shown in Table 7.12. The search times for seven year old children are not shown because this measure cannot be obtained from the videotapes.

Table 7.12: Time spent searching when producing the first two layers of the Tower, for the original model and the RT6 model. Standard deviations are shown in parentheses.

	Original model	RT6 model
First layer (size6)	9.0 s (3.9)	11.6 s (7.4)
Second layer (size5)	6.5 s (4.5)	11.6 s (8.6)

There are no reliable differences between the models for the time spent searching when producing the first layer ($t(18)=0.99$, $p>0.05$) or the second layer ($t(18)=1.66$, $p>0.05$). There is an interaction between memory and errors, however, because the more errors that are made, the more load there is on memory (because time increases which means decay has more influence). This explains the high variance on times for the RT6 model. The importance of memory capacity increases as the number of errors increase.

As with the strategy choice modification, lower levels of retrieval threshold need to be examined to determine whether they show the same pattern of behaviour as the RT6 model. This is because when several modifications are implemented simultaneously, lower levels of retrieval threshold will be used. The threshold models RT2-RT5 have an average correlation with the RT6 model of 0.81 for the number of construction attempts produced for each layer, and 0.87 for the time taken to construct each layer. The lower levels of threshold show a similar pattern of behaviour to the RT6 model.

The M-power modification has shed light on the importance of memory capacity, and has shown that memory capacity can shift the behaviour of the model towards the seven year old children. However, the modification did not mean the RT6 model's behaviour differed significantly from the original model.

7.2.2.2 Case's mental capacity

Case (1985) proposed a functional capacity limitation whereby capacity has the same limit across ages. The capacity limit is the amount of memory elements that can be processed simultaneously. Older children can manipulate the limitation better than younger children because they store information more efficiently. Efficient storage of information (e.g. by chunking knowledge, or automising processes) means that the

information requires less memory elements. Information that is stored efficiently therefore uses less capacity than information that is stored inefficiently.

The simultaneous processing of memory elements in the model is represented by the number of conditions that can be matched in production rules. Figure 7o shows the distribution of the number of conditions in production rules. The highest is twelve. If a higher functional capacity can alternatively be represented as a higher number of conditions that are allowed in productions, then lowering the maximum number of conditions in a production rule will examine the lower functional capacity that seven year old children are hypothesised to have.

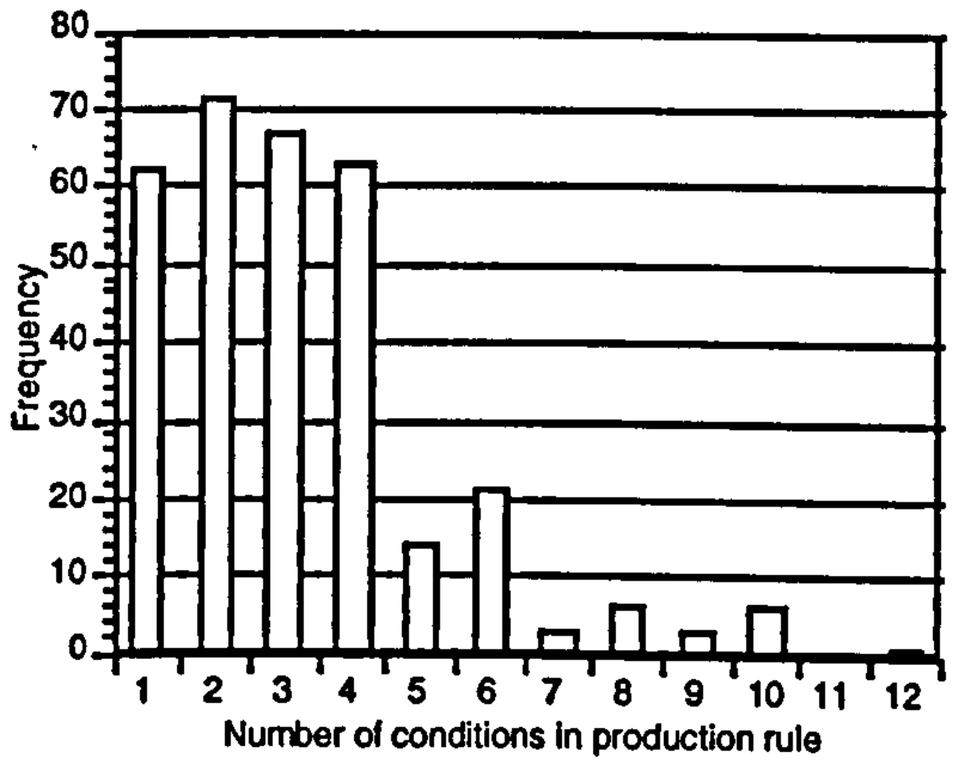


Figure 7o: Frequency in the model of production rules having different numbers of conditions.

The role of automisation is ignored in the modification to the model, because the model will be modified to represent the capacity *before* automisation of processes occurs, with the original model representing the state of cognition *after* automisation. Combining DME's through practice is one method by which the model would require less conditions in productions, although this mechanism is not implemented in the model as yet.

There are three methods by which the number of conditions in productions can be reduced: ignoring the production altogether; splitting the production into two or more sequential productions; chunking the knowledge (i.e. the conditions) such that the same information can be represented by less conditions. The first two will be examined (as stated above, the model is not yet capable of chunking knowledge automatically).

To execute this modification, every production that exceeds the limit of conditions in a production rule is split into two productions. Where a production cannot be split, it is

ignored. All of the production rules in the model can be split, except the production rules that require twelve conditions. These rules carry out reflection of the task (creating new reflection rules), and co-ordinate two pairs of features. The new rule is based on the co-ordination of the features and is difficult to split into two or more rules: the reflection rules are therefore omitted.

The limit of conditions in a production will be varied from six to ten. The limit of ten conditions per production means the model proceeds as normal, but ignores reflection. All other limits on conditions mean one or more productions being split into two:

1. Nine condition limit: Four productions in the Fit Together module were split into two sets of productions, four having seven conditions and four having six conditions.
2. Eight condition limit: The nine condition limit model was used, meaning three productions in the Decide Fit module were split into two sets of productions, three having eight conditions and three having three conditions.
3. Seven condition limit: The eight condition limit model was used, meaning three productions in the Decide Fit module were split into two sets of productions, three having six conditions and three having three conditions.
4. Six condition limit: The seven condition model was used, meaning four productions in the Fit Together module were split into two sets of productions, four having six conditions and four having two conditions. Three productions in the Decide Search module were also split into two sets of productions, three having five conditions and three having three conditions.

Figure 7p shows the time taken for each of the limited condition models. The limit of six conditions in a production is not shown because the timings are significantly longer than for the seven condition limit (three runs of the six condition model averaged ten minutes simulation time to complete the Tower; as each run took four hours in real-time the remainder of the ten runs were omitted).

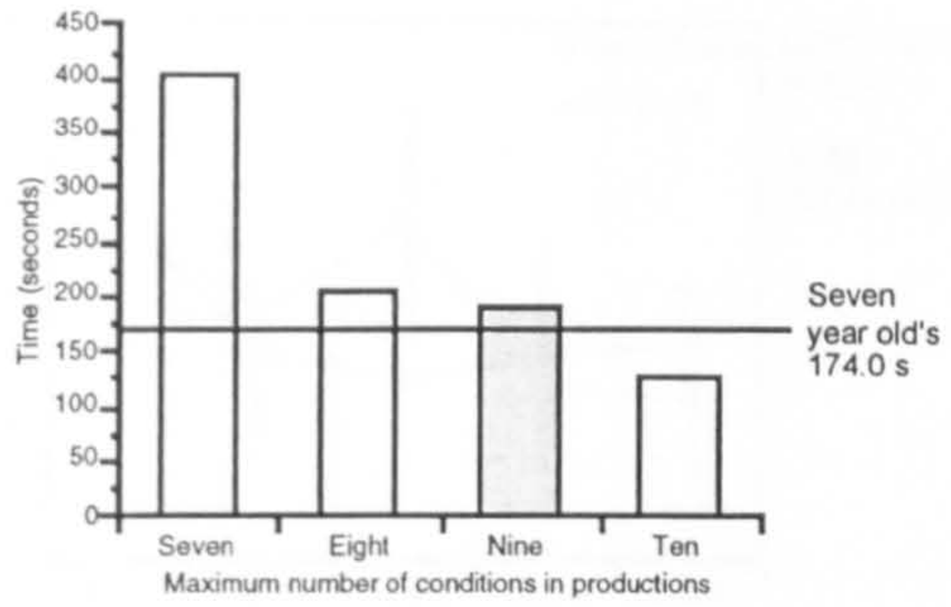


Figure 7p: The time taken by the model for different levels of condition limits in productions.

The model is closest to the time taken by seven year old's when the limit on conditions in productions is nine (henceforth the C9 model). This model will therefore be examined in more detail. Table 7.13 shows comparisons across models and the seven year old children, for construction attempts and time taken.

Table 7.13: Construction attempts and time taken for the original model, the C9 model, and seven year old children. Standard deviations are shown in parentheses.

	Original model	C9 model	Seven year old children
Construction attempts	23.1 (2.4)	29.6 (4.9)	27.6 (3.5)
Time taken	129.0 s (31.5)	188.2 s (41.5)	174.0 s (32.1)

There is a significant difference between the original and C9 models for both construction attempts ($t(18)=3.77$, $p<0.01$) and time taken ($t(18)=3.59$, $p<0.01$). The C9 model is within 20% of the seven year old children's scores for both time taken and the number of construction attempts. The measures will be examined further by looking at them on a layer-by-layer basis. Figures 7q and 7r show the time taken to construct each layer, and the number of construction attempts made in producing each layer, for the C9 model and seven year old children. The C9 model does not correlate well with seven year old's for either measure. There is a large average error (indicated by RMS error) between the C9 model and the seven year old's for both layer timings and number of construction attempts per layer.

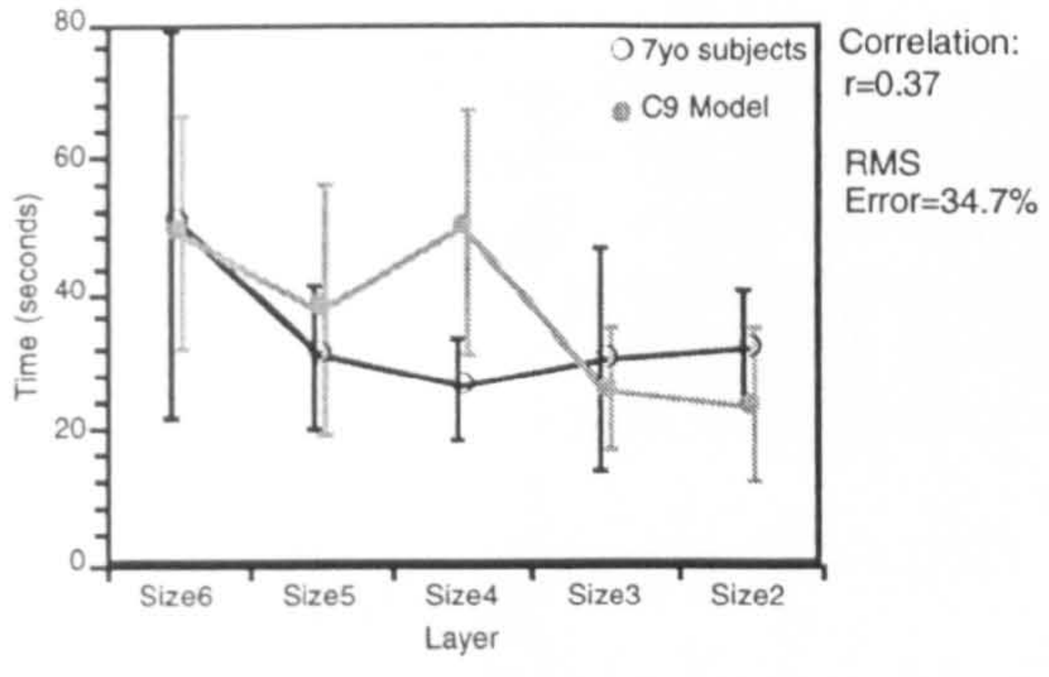


Figure 7q: Time taken to complete each of the five layers, for the C9 model, and the seven year old subjects.

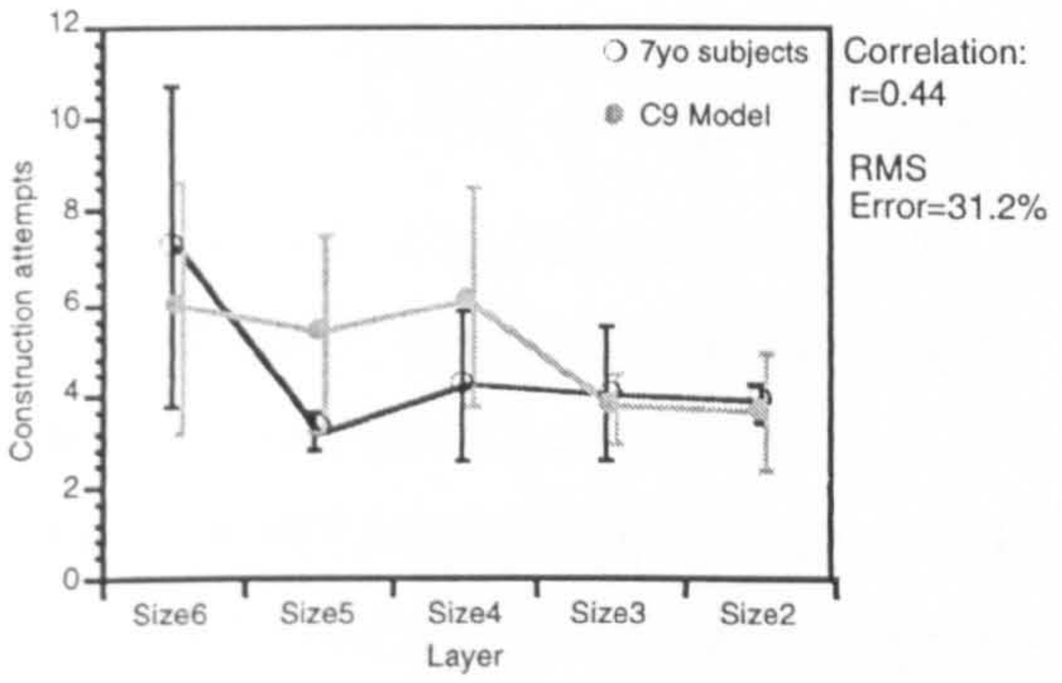


Figure 7r: Constructions made in completing each of the five layers, for the C9 model, and the seven year old subjects.

If the average times to complete the size4 layer were excluded, the layer timings for the C9 model would correlate very well with the seven year old's. The number of construction attempts is also high for this layer. There are no reliable differences between the C9 model's layer timings for the first three layers. It is possible that splitting the productions has slowed the learning mechanism of the model.

Figure 7s shows the expected gain values for productions in a sample run of the C9 model. The area of interest is the constructing of the size4 layer, so the graph begins at the start of the construction of the size5 layer and ends when the size3 layer has been constructed. The sample run takes 57 seconds to build the size5 layer, 57 seconds to build the size4 layer, and 32 seconds to build the size3 layer. Productions are only graphed if they fire in the time span of the graph (quarter circle alignment productions are not shown for this reason). When a production fires, its expected gain value normally changes, which is seen on the graph as the value rising or falling.

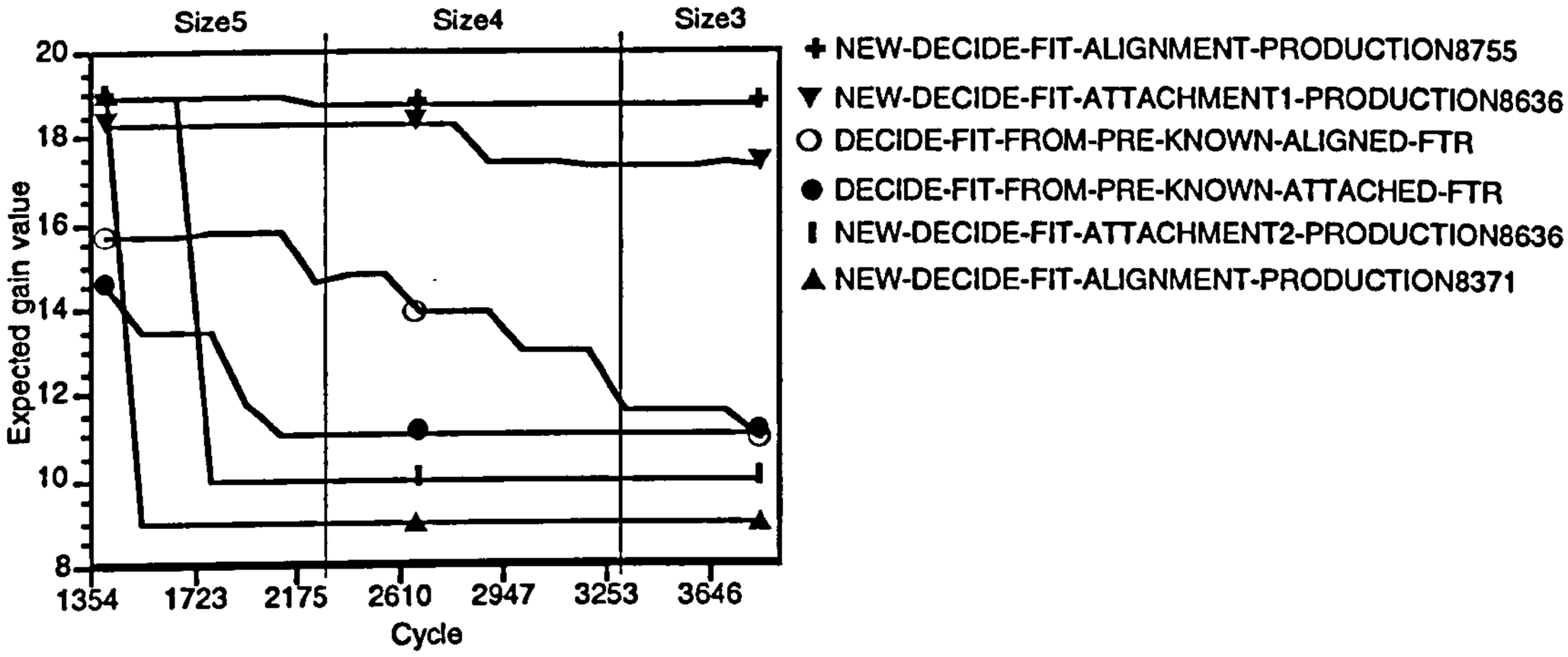


Figure 7s: Expected gain values of productions of a sample run of the C9 model. The graph begins at the start of making the size5 layer and ends when the size3 layer is constructed.

The sample run shown in Figure 7s is split into three areas, depicting when the model is constructing the size5 layer, the size4 layer, and the size3 layer. All of the competing productions are either to attach two features or to align two features (which are not quarter circles). The productions that are created during the model run (prefixed by "new-") are for *specific* features, whereas the pre-known productions are general. This means that the pre-known productions may be matched in situations when the new productions cannot.

The pre-known-aligned production is fired steadily throughout the building of the size5 and size4 layers, until its expected gain value becomes so low that even when taking into account the noise placed on expected gain values, the production cannot fire. The pre-known-aligned production would seem to be a source of the extra time taken, because it is fired several times during the construction of the size4 layer. However, alignment productions always result in success (given blocks of the same size). The expected gain value reduces with each firing of the production, so the cost of achieving success (i.e. the time taken to find and fit blocks) must outweigh the benefits of achieving the success.

Analysing the firing of the production shows the added cost occurs because the model often has to fixate on the blocks involved in the fit. Limiting the number of conditions in productions meant splitting four productions in the Fit-Together module for the C9 model. This causes four productions to fire for every Fit Together in the C9 model, instead of two (for the original model). Decay occurs every cycle, so more production firings means more decay. This is the likely source of the extra fixations for the size4 layer. The C9 model spends significantly more time on eye movements and fixations (9.9

s) than the original model does (4.1 s) when constructing the size4 layer ($t(18)=3.73$, $p<0.01$), but not for any other layers. The extra decay is more significant for the size4 layer.

7.2.2.3 Halford's capacity limit

Halford (1993) proposed that capacity increased with age. The main area where capacity limitations would influence performance was when parallel processing was required, because parallel processing requires several things to be considered at once.

In the model, each rule has a number of conditions that must be satisfied in order for the rule to be placed in the conflict set (the set of rules from which a rule is selected to fire). Satisfying each condition within a rule is a parallel process. Rules with many conditions to be satisfied should place more exertion on capacity than rules with relatively few conditions to be satisfied, because they require more conditions to be satisfied at once. Therefore manipulating the number of conditions that can be satisfied at once, for any one rule, should enable Halford's capacity limitation theory to be examined.

This modification is the same as the one used to operationalise the capacity limitations proposed by Case. However, Halford suggests three alternatives for dealing with limited capacity: (a) decompose the task into smaller segments (which are processed serially); (b) recode the representation; (c) default to an easier strategy or concept. Alternatives (a) and (b) have been considered when examining Case's capacity limitation theory (although re-representation was discarded). Alternative (c), defaulting to an easier concept, can only be examined if there is an easier concept to default to. In the model this means there must be less complex productions that accomplish the same task as more complex productions. This is the same as using productions with a small number of conditions to perform the same task as productions with a large number of conditions. This has already been explored with the Case modification. Therefore Halford's theory was tested with the modifications for Case's theory, and will not be reported here.

7.2.3 Modifications to processing speed

Three areas where speed may differ across ages were identified in Chapter 2: processing speed, eye movement (and possibly fixation) speed, and the speed of hand movements. Each of the three will be modified in the model. For each modification, the construction behaviour will not change; differences will only be seen for the overall task time, and the time taken to produce each layer. The correlation for layer timings between the model and the seven year old children will not alter, because timings will increase uniformly across

layers. The effect of speed will therefore be examined using the RMS error between each layer timing for the modified model and the seven year old's. This will indicate the percentage time difference between the model and seven year old's for producing a layer.

Increasing the speed of cognition, eye movements and fixations, and hand manipulations will at some point match the task timings of seven year old children. Modifications to processing speed are made in order to see to what extent each level of speed alters the timings of the model. Processing speed is a theoretically justifiable modification but will need to be used in conjunction with other modifications that change the construction behaviour of the model.

7.2.3.1 Cognitive speed

The cognitive processing speed in the model is set by the default action time parameter, and is the time per production rule firing. The default action time was set to 50 ms for the original model (see Chapter 6). This time was increased, ranging from 65 ms to 105 ms, in units of 10 ms. Table 7.14 shows the RMS error for layer timings between each level of cognitive speed and the seven year old children.

Table 7.14: RMS error for layer timings for the different levels of cognition timing and the seven year old children.

Cognition timing	65 ms	75 ms	85 ms	95 ms	105 ms
RMS error (%)	20.2	16.3	15.4	17.8	22.2

Table 7.14 shows that a cognitive processing speed of 85 ms is the best approximation to the seven year old data. At this speed, the overall task time is 171.0 s; the seven year old children take 174.0 s (the modified model is therefore within 20% of the subject score).

7.2.3.2 Motor action speed

Differences in motor action speed will be examined in the same way as the differences in processing speed, because it is only the model layer timings and overall timings that will change. There are two types of motor action in the model: eye movements and fixations, and hand movements.

Eye movements and fixations

The eye movement time is set to 50 ms in the model, and the fixation time is set to 200 ms. For every increment in eye movement time, the increment in fixation time should be increased fourfold (assuming eye movement time and fixation times increase linearly). Eye movement timings were set from 60 ms to 100 ms in increments of 10 ms. The

fixation times therefore varied from 240 ms to 400 ms. Table 7.15 shows the RMS error for layer timings between each level of eye movement and fixation speed, and the seven year old children.

Table 7.15: RMS error for layer timings for the different levels of eye movement and fixation speed, and the seven year old children.

Total time for eye movement and fixation	300 ms	350 ms	400 ms	450 ms	500 ms
RMS error (%)	25.8	22.5	20.5	18.1	16.3

Table 7.15 shows that an eye movement time of 100 ms and a fixation time of 400 ms is the best approximation to the seven year old data. At this speed, the overall task time is 156.5 s; the seven year old children take 174.0 s. The modified model is therefore within 20% of the subject score. Higher timings for eye movements and fixations would further reduce the RMS error. Timings above the 500 ms maximum used here were considered unrealistic because it would represent such a slow eye movement/fixation.

Hand movements

Timings relating to hand movements and block manipulations are set to 550 ms in the model. Hand movement and block manipulation timings were increased from 650 ms to 1050 ms in increments of 100 ms. The maximum of 1050 ms is realistic based on estimates from the video's of seven year old children completing the task (the videos indicate a hand movement speed averaging 850 ms, but the range from 650 ms to 1050 ms will be used to examine this speed further). Table 7.16 shows the RMS error for layer timings between each level of eye movement and fixation speed, and the seven year old children.

Table 7.16: RMS error for layer timings for the different levels of hand movement timing, and the seven year old children.

Hand movement timing	650 ms	750 ms	850 ms	950 ms	1050 ms
RMS error (%)	24.9	21.3	18.1	15.4	13.3

Table 7.16 shows that a hand movement speed of 1050 ms is the best approximation to the seven year old data. At this speed, the overall task time is 164.0 s; the seven year old children take 174.0 s. The modified model is therefore within 20% of the subject score.

7.2.4 Summary of the modifications

Table 7.17 shows a summary of the effect the modifications have had on the model's behaviour. The "Effect on behaviour" column indicates how the modification changed the behaviour of the model. The fraction shows how many of the modified model's scores are within 20% of the seven year old children's scores on the nine measures shown in Figure 7a. The time taken and number of construction attempts are also given separately (the two measures are the most important of the nine). An upward arrow indicates the score changed to be within 20% of the score for seven year old children. An equal sign the score did not change. Where layer correlations are stated, the correlation is 0.70 or more.

Table 7.17: Summary of how well the modified model's behaviour matched the behaviour of seven year old children on the Tower task.

What develops	Method of operationalisation in model	Effect on behaviour
Internalising external operations	Removal of internal flush edges check	**3/9** **Time =** **Constructions =**
	Removal of internal feature obstruction check	**1/9** **Time =** **Constructions =**
Strategy choice	Increase in expected gain noise	**6/9** **Time ↑** **Constructions ↑** **Layer times correlate** **Constr. attempts correlate**
Strategy efficiency	Reduced accuracy of internal flush edges check	**7/9** **Time ↑** **Constructions ↑** **Layer times correlate**
	Reduced accuracy of internal obstruction check	**4/9** **Time =** **Constructions =**
Procedural knowledge	Removal of strategy for remembering blocks seen	**3/9** **Time =** **Constructions =**
	Removal of strategy for remembering fit attempts	**0/9** **Time =** **Constructions =**
	Removal of search by features	**3/9** **Time =** **Constructions =**

What evolves	Method of operationalisation in model	Effect on behaviour
Declarative knowledge	Removal of knowledge that halfpegs align, and halfholes align	**2/9** **Time =** **Constructions =**
	Removal of knowledge that halfpegs fit into halfholes	**3/9** **Time =** **Constructions =**
	Removal of knowledge that quarter circles align	**2/9** **Time =** **Constructions =**
Capacity	Increasing retrieval threshold	**5/9** **Time ↑** **Constructions ↑**
	Reducing the number of conditions that can be matched in each rule	**4/9** **Time ↑** **Constructions ↑**
Processing speed	Increased cognition timing	**3/9** **Time ↑** **Constructions =**
	Increased eye movement and fixation timings	**3/9** **Time ↑** **Constructions =**
	Increased hand movement timings	**3/9** **Time ↑** **Constructions =**

Table 7.17 shows that some modifications have led to dramatic changes in the model's behaviour, yet some have had no effect on behaviour at all. The chapter has shown that *mechanisms of development that are proposed by developmental theories can be implemented within a single computational model.* The results of the implementations show that the mechanisms can alter the behaviour of the model, but to varying degrees. This constitutes an important finding in relation to testing mechanisms of development:

the influence of each mechanism on task behaviour can be elicited (as shown by Table 7.17).

7.3 Summary

The summary of the modifications will be in four parts. First, the fact that some modifications had no effect on behaviour is discussed. Second and third, the implications of the results of some of the modifications for developmental theory and models of development is discussed. Fourth, an overall summary is given.

7.3.1 Why do some mechanisms have no effect on the model's behaviour?

Some mechanisms have failed to alter the behaviour of the model. The reason for this could be how the mechanisms were implemented in the model. The mapping between modifications and developmental mechanisms proposed by theories should be very good. Where this is not the case the fault lies either with the theories of development (being imprecise in detailing mechanisms) or with the implementation of the mechanism within a computational framework. One particular ill-defined mechanism is knowledge because many of the theories are not precise in how knowledge is represented, and in how knowledge changes. All of the modifications that failed to change the model's behaviour were knowledge changes. As the knowledge modifications were easy to implement within ACT-R (the ACT-R knowledge base is well defined), the operationalisation of the mechanism is not the reason for it having no effect on behaviour. Two other possibilities stand out:

1. The representation of the task is incorrect. This is entirely possible and is a common criticism of computational modelling (e.g. Searle, 1980/1997). However, the physical nature of the task reduces the degrees of freedom in selection of the declarative knowledge which the model uses. The main criticism has to be upon the procedural knowledge. A good fit between the original model and the adult subjects on time taken (overall and per layer) would suggest that the degrees of freedom have been reduced for procedural knowledge also. However, the problem exists and must be borne in mind.
2. The selection of which knowledge to remove was incorrect. However, the removed knowledge was from the central behaviour modules and therefore should have the highest influence on behaviour.

The knowledge modifications had no effect on behaviour either because the representation of the task was incorrect, or the knowledge removed doesn't influence behaviour. Every

other mechanism besides knowledge led to a change in the model's behaviour, suggesting that theoretical mechanisms of development can be implemented successfully within a computational framework.

7.3.2 Implications for theories of development

The knowledge modifications which did not affect behaviour are interesting, because some of them were implemented in more than one way. The internal fitting procedure was removed as one modification, and had its accuracy altered with another modification. Removing the knowledge did not change the model's behaviour, yet changing the accuracy did. This shows the problems in studying development because it shows that performance in some cases is not hindered by a lack of knowledge, but is in fact hindered by the lack of experience in using the knowledge (assuming that the accuracy of the knowledge [e.g. a strategy] improves with use).

In general, the modifications have shown that many theories need to think of knowledge in a different way. It is not just basic knowledge (in terms of what could be called facts and rules within a computational framework) that is changing. The knowledge is more elaborated than this – it can be mediated by capacity, and can have an associated accuracy, to name two examples. This is an important finding because it suggests (as Siegler's theory also indicates) that knowledge within theories of development needs to be further defined. Siegler has already proposed various subsets of knowledge that affect children's behaviour. However, he does not propose methods by which the basic knowledge is acquired. New theories of development need to cover both knowledge acquisition and define in detail the aspects that influence knowledge use (e.g. accuracy).

7.3.3 Implications for models of development

None of the individual modifications presented altered the model's behaviour enough to match the behaviour of seven year old children on the Tower task (although micro-levels of some parameters were not pursued). The best fit on the nine measures given in Chapter 4 was 7/9. This is somewhat surprising because some of the models presented in Chapter 3 were able to match subject data by making individual modifications (knowledge) to the model. Two reasons can be put forward as to why other models are able to fit the data by making individual modifications:

1. In the domain modelled, individual modifications are adequate. This is supported by Gentner, Rattermann, Markman and Kotovsky (1995), but they also accept that there are other mechanisms of development than knowledge.

2. A small range of measures are used to match the subject data and so the mismatches are not seen. This criticism has been covered in Chapter 3. It is supported here by most of the modifications managing to match subjects on at least two or three measures – if these had been the only measures used to match subject behaviour, many of the modifications would have been considered successful.

Modifications other than knowledge have changed the behaviour of the model. Current models of development only allow for knowledge modifications. The implication for models of development is that other mechanisms need to be considered when modelling developmental phenomena in specific tasks. In particular, the modifications have highlighted the areas which transition mechanisms need to be aware of (e.g. accuracy, capacity, processing speed). Any transition mechanism will have to include some hypotheses about how processes such as capacity and processing speed interact.

7.3.4 Overall summary

Some of the individual modifications have changed the model's behaviour so that it behaves more like seven year old children. However, the modifications have not been able to fit the behaviour seven year olds on every measure. This suggests that for the Tower task, either other types of modification need to be introduced, or the modifications presented need to be interacted with one another. Combining modifications supports the majority of theories of development which each present several mechanisms by which development occurs. Chapter 8 will examine the effects of combinations of mechanisms on the model's behaviour.

8. Combining modifications to the model

Chapter 7 showed that mechanisms of development that are proposed by theories of development can be implemented and tested within a computational model. Some of the mechanisms substantially altered the behaviour of the model. Many theories of development propose multiple mechanisms that influence development. This chapter examines how combinations of modifications can change the behaviour of the model, and proposes a theory of how transition mechanisms can be implemented within the ACT-R architecture.

8.1 Combined modifications to the model

Chapter 7 presented seventeen different modifications to the Tower model. The first part of this section discusses why it is unnecessary to examine every combination of the seventeen modifications. The second part discusses which of the seventeen modifications to pursue when making combinations. The third part summarises the results of the combined modifications when the settings for each modification are the same as they were in Chapter 7 (remember some modifications involved variable parameters with multiple settings). The fourth part varies the settings for modifications and summarises the best results from the combined modifications. The fifth part compares the results of the best combination of modifications with the best individual modification from Chapter 7.

8.1.1 Does every combination of modifications need to be examined?

Most verbal theories which include several mechanisms of development also include hypotheses about the ways in which the mechanisms interact (e.g. Case, 1985). The ACT-R architecture is a general architecture which was designed to be a unified theory of human cognition. The architecture was not designed to incorporate specific mechanisms of development. Although Chapter 7 showed that some developmental mechanisms can be mapped onto some of the mechanisms that are already in the ACT-R architecture, there can be no architectural theory regarding how all the developmental mechanisms that have been implemented interact with each other (because not all of the mechanisms are ACT-R mechanisms).

When a theory exists which describes how different mechanisms interact with each other, then predictions can be made regarding what behaviour should be seen. As there is no theory regarding how ACT-R mechanisms interact with external mechanisms, then the behaviour of an ACT-R model cannot be predicted beforehand if it uses a combination of

internal and external mechanisms (actually, it is nearly impossible to make predictions even when combining ACT-R specific mechanisms). The behaviour of the model, for each combination of mechanisms (or modifications, as one modification corresponds to a developmental mechanism), has to be found by running the model. Given the random aspects of the Tower model (see Chapter 6), this means that every combination of modifications will require at least ten runs of the model.

To find the best possible combination of modifications (the combination that provides the best match to the data), all of the possible combinations of modifications have to be examined. This sets up a search space containing every possible combination. Traversing the search space will be unwieldy and time consuming, because it is so large, and because testing each combination requires ten runs of the model. It is therefore unwise (but not impossible) to examine every possible combination of modifications.

Within the bounds of this thesis there is a more fundamental reason for not examining every combination of modifications: it is not necessary. Chapter 7 showed that mechanisms of development can be implemented and tested within a computational framework. The combinations of modifications are examined because some theories explain development using several mechanisms, suggesting that only a combination of mechanisms can explain what develops. The combined modifications tested here are needed to show that combinations can be achieved within a single model (i.e. that the search space can be traversed), and that the combinations result in the model's behaviour moving toward the behaviour of seven year old children. The combinations do not yet need to provide a perfect match to the seven year old data (although they should suggest that this *is* possible with future work), and therefore every possible set of combinations need not be pursued.

8.1.2 Which combinations to pursue

Several of the modifications presented in Chapter 7 moved the behaviour of the Tower model to be more like the behaviour of seven year old children. In particular, four modifications matched children on both of the main two measures of behaviour (construction attempts and time taken):

- Increase in expected gain noise (the EGN6 model)
- Reduced accuracy of internal fitting (the Flush40 model)
- Increasing retrieval threshold (the RT6 model)

- Reducing the number of conditions that can be matched in each rule (the C9 model)

It would seem sensible to select the four modifications which best matched the behaviour of children on the central behavioural measures, and so these four are selected for combining. The other modifications may influence behaviour when they are combined, but just these four will be considered.

The four modifications listed above will be examined by combining them into pairs, giving six sets of paired modifications to make to the model. One combination of modifications (retrieval threshold and conditions in productions) examine the same developmental mechanism (capacity), but other combinations are proposed by single theories of development (e.g. expected gain noise [strategy choice] and accuracy of internal fitting [strategy accuracy] are both proposed by Siegler). Combinations which are not proposed by developmental theories signify the creation of new theories.

Three of the four modifications (expected gain noise, reduced accuracy of internal fitting, and retrieval threshold) that have been selected for combining are not simple present-or-absent modifications – they have a range of parameter settings. The parameter settings therefore increase the search space again, because three of the modifications can have a range of settings, each of which needs to be examined. Before examining a range of parameter settings, a summary of the model's behaviour for the combined modifications using the parameter settings from Chapter 7 is given.

8.1.3 Results of combined modifications using Chapter 7 settings

The model's behaviour for construction attempts and time taken for each of the six paired combinations of modifications is shown in Table 8.1. The scores on these measures for seven year old children is also shown. The parameter setting (where applicable) is the optimal setting as found in Chapter 7. The key for the combined modification models and parameter settings is given in the table header.

Table 8.1: Summary of construction attempts and time taken with paired combinations of modifications, keeping parameter settings as they were in Chapter 7. Key to each modification type: C9=Rules limited to nine conditions; EGN6=Expected gain noise was set to 6; RT6=Retrieval threshold was set to 6; Flush40=Probability of an internal flush edges check being incorrect was set to 40%.

Modification combination	Construction attempts	Time taken
Seven year old's	27.6	174.0
C9 and EGN6	32.3	221.6
C9 and RT6	25.5	184.3
C9 and Flush40	33.1	228.6
Flush40 and EGN6	25.9	209.6
Flush40 and RT6	26.3	215.6
RT6 and EGN6	27.1	217.3

Initial testing of the pairs of modifications using the settings from Chapter 7 shows that the paired modifications change the behaviour of the model too much. Most of the paired modifications lead to the model performing worse than seven year olds for either construction attempts, time taken, or both. This shows that either the modifications interact with each other or the sum of the two modifications has a greater effect on behaviour than each individual modification.

To be successful, the paired modifications need less extreme parameter settings. A distribution of parameter settings were examined. The next section shows the *best* combinations of parameter settings. The settings were not optimised (as this would involve too large a search space) so it is possible that better results could be obtained with different parameter settings.

8.1.4 Results of combined modifications using variable settings

Each modification that involved the setting of a parameter was given a distribution of parameter settings to combine with the other modifications. This resulted in a large number of paired combinations of modifications. Table 8.2 shows the best results for each combination of modifications. The key to each combination of modifications, and the parameter setting for each modification, is given in the table header. The first results column shows how many of the principle nine task measures were within 20% of scores

for seven year old's. The second and third results columns show the correlations and RMS error values for the within-task measures.

Table 8.2: Summary of the results of the paired combinations of modifications. Key to each modification type: C9=Rules limited to nine conditions; EGN2,3.5,4=Expected gain noise was set to 2,3.5, and 4; RT2,5=Retrieval threshold was set to 2 and 5; Flush8,15,35=Probability of an internal flush edges check being incorrect was set to 8%, 15%, and 35%.

Combination	Number of overall measures within 20%	Construction attempts per layer	Time taken per layer
Number of conditions and strategy choice (C9–EGN2)	4 / 9	r = 0.66 RMS error=24.6%	r = 0.81 RMS error=15.1%
Number of conditions and retrieval threshold (C9–RT5)	7 / 9	r = 0.25 RMS error=49.0%	r = 0.12 RMS error=43.0%
Number of conditions and strategy accuracy (C9–Flush8)	5 / 9	r = 0.63 RMS error=26.8%	r = 0.62 RMS error=24.3%
Strategy accuracy and strategy choice (Flush15–EGN3.5)	4 / 9	r = 0.76 RMS error=29.0%	r = 0.86 RMS error=29.1%
Retrieval threshold and strategy accuracy (RT2–Flush35)	5 / 9	r = 0.77 RMS error=22.3%	r = 0.97 RMS error=24.9%
Retrieval threshold and strategy choice (RT2–EGN4)	3 / 9	r = 0.91 RMS error=26.8%	r = 0.72 RMS error=19.9%

On average, the paired combinations of modifications enable the model to match subjects better than the individual modifications (an average of 4.67 measures match subjects for the combined modifications compared to an average of 3.25 for the individual modifications). Given the correlation criteria applied in Chapter 7 (where a successful correlation between model and subjects was a correlation of $r \geq 0.70$), the final three paired combinations correlate on both construction attempts and time taken per layer, although the error for these is high.

8.1.5 Comparison of the individual and combined modifications

There are two types of measures for which the combined modification and individual modification models can be compared on: the nine principle measures (i.e. overall measures) and the within-task measures (see Chapter 4). The models which best match subjects for overall measures and within-task measures will be compared.

8.1.5.1 Overall measures

For the nine principle measures of behaviour (see Chapter 4), the best of the combined modifications is the C9–RT5 model which matches the seven year old children on seven out of the nine measures. The best model presented in Chapter 7 was for the accuracy of the internal fitting modification (the Flush40 model) which also matches on seven out of nine measures. Table 8.3 shows a comparison between seven year old children, the C9–RT5 model, and the Flush40 model, for scores on the principle nine measures.

Table 8.3: Summary of scores on the principle measures of behaviour on the Tower task, for seven year old children, the C9–RT5 model, and the Flush40 model. Model scores within 20% of subject scores are indicated with a *. Standard deviations are indicated in parentheses.

	Seven year old children	C9–RT5 Model	Flush40 Model
Construction attempts[5]	27.6 (3.5)	29.8* (2.9)	26.1* (3.0)
Time taken	174.0 s (32.1)	175.0 s* (17.0)	183.3 s* (35.9)
Incorrect constructions involving blocks of the same size	5.4 (2.7)	8.3 (2.6)	5.2* (2.6)
Incorrect constructions involving blocks of different sizes	1.8 (2.7)	0.0	0.0
Average number of times the same blocks are fit together in the same way again	1.0 (0.4)	1.1* (0.3)	1.0* (0.0)
Average number of times the same blocks are fit together in different ways	1.2 (0.5)	1.2* (0.5)	1.2* (0.6)
Time between correct constructions	8.5 s (10.8)	8.1 s* (8.6)	8.7 s* (9.3)
Time between incorrect constructions	2.9 s (6.3)	2.9 s* (6.3)	3.2 s* (5.8)
Number of task appropriate features	1.3 (0.6)	1.5* (0.5)	1.8 (0.5)

The C9–RT5 model differs from the Flush40 model in two respects. First, the C9–RT5 model matches subjects for the number of task appropriate features for incorrect constructions (i.e. how many features are attached or aligned correctly even though the construction itself is incorrect), which the Flush40 model fails to do. Second, the C9–RT5 model fails to match subjects for the number of incorrect constructions involving blocks of the same size, which the Flush40 model manages to do.

---

5 The number of correct constructions produced can be more than twenty (the optimal amount).

The C9–RT5 model manages to match subjects for task appropriate features because the limit on conditions in its productions means that features are co-ordinated less often, leading to less features being attached or aligned correctly. This would seem to be an important aspect in matching the data from seven year old children because the children's errors do not include many features that are attached and/or aligned in a correct manner.

The Flush40 model manages to match subjects for the number of incorrect constructions involving blocks of the same size because it produces relatively few errors overall. The errors by seven year old children involve both same-size and different-size blocks. The Flush40 model produces less overall errors than seven year old children, and this is why its same-size errors match those of the seven year old children. The number of errors involving blocks of different sizes is an important measure because it differs across children and adults. Any further models must try and incorporate errors involving blocks of different sizes.

8.1.5.2 Within-task measures

For within-task measures (layer-by-layer correlations between model and subjects for time taken and construction attempts), the RT2–Flush35 model is the best of the combined modifications (r=0.97 for time taken and r=0.77 for construction attempts), and the EGN6 model is the best of the individual modifications (r=0.99 and r=0.73 respectively). Figures 8a and 8b show the layer-by-layer comparisons for the seven year old children, the RT2–Flush35 model, and the EGN6 model (for clarity, error bars are omitted).

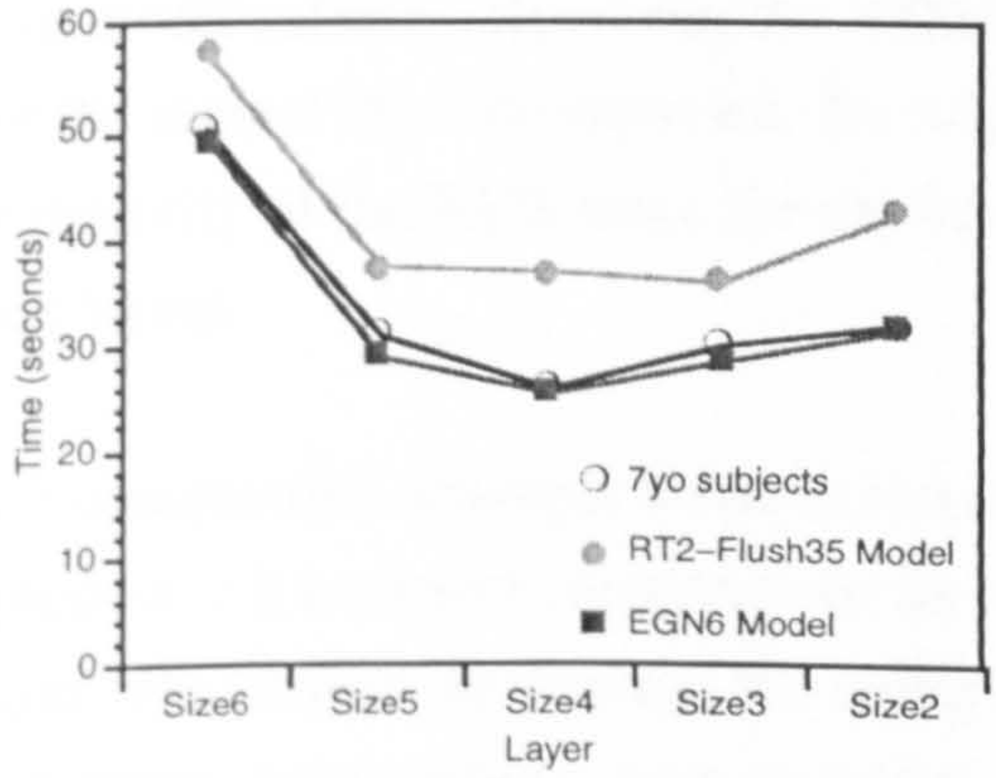


Figure 8a: Time taken to complete each of the five layers, for the seven year old children, the RT2–Flush35 model, and the EGN6 model.

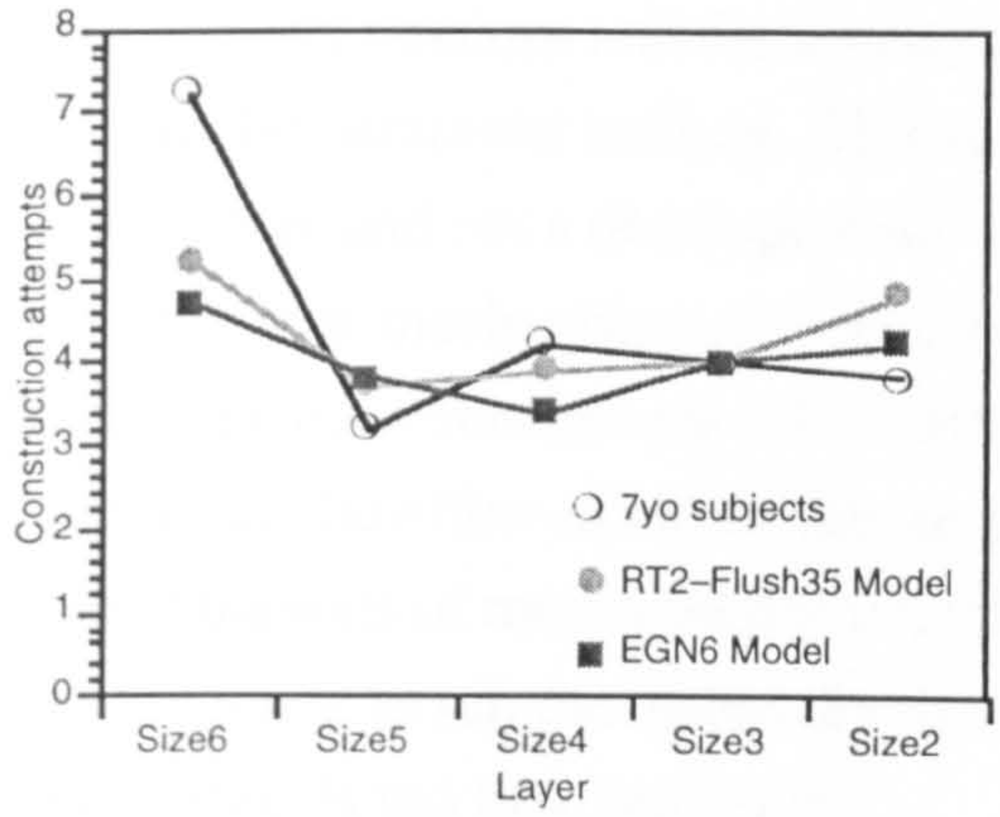


Figure 8b: Constructions made in completing each of the five layers, for the seven year old children, the RT2–Flush35 model, and the EGN6 model.

Both models correlate well for time taken, although the EGN6 model has a far superior RMS error than the RT2–Flush35 model (3.0% versus 24.9%). The models also correlate for construction attempts, but the RMS error is poor for both (26.8% and 22.3%). The EGN6 model provides a better overall match to subjects for within-task measures than the RT2–Flush35 model. The RT2–Flush35 model should provide a better match when its parameter settings are optimised.

Almost all of the combined modifications show a high RMS error for both construction attempts and time taken. The majority of the RMS error for construction attempts occurs when constructing the first layer where seven year old children make 7.2 construction attempts (the most that any combined modification model makes is 5.3; a similar result is also found for the individual modifications). However, the children learn from the errors made on the first layer (for the second layer constructed, the number of construction attempts falls to 3.2). The majority of the RMS error for the time taken tends to be spread across the first three layers.

The difference in first layer construction attempts between the subjects and the models is because seven year old's produce 1.8 incorrect constructions involving blocks of different sizes when making this layer (see Chapter 4) whereas the majority of the models produce none. If the different-size incorrect constructions were excluded, the models would match the children's data well. The selection of incorrect size blocks is a modification to the model that has not yet been examined. If the goal is to fully match the seven year old children's data, then future work must look at why incorrect sized blocks are selected.

The match between the combined modifications and the seven year old children's data could be improved by optimising the parameter settings. This must be left for future work. ACT-R is a general architecture and not a developmental architecture. This means ACT-R makes no predictions as to how mechanisms such as capacity should change, and what effect the changes will have on other mechanisms. The lack of specific predictions means the model has to be run to see the effect of combinations of modifications. The search space of all possible combinations of modifications (and their parameter settings, if applicable) is large and the requirement to run the model ten times for every combination means that testing every combination is too time intensive.

The above results represent the first steps in combining modifications. The results show that the Tower model can be used to routinely create and test a large number of theories of development. Current theories can be implemented and tested within a computational framework. Novel theories (i.e. combinations of modifications that have not been described before) can easily be created and tested within a computational framework.

This section has also highlighted the need for a coherent theory for how different developmental mechanisms interact. The mechanisms of development can indicate *what develops*, but something else is required to state *how the development takes place* (i.e. how the mechanisms interact). The next section details a theory of how the different developmental mechanisms could interact within the ACT-R architecture. The section details a transition mechanism which brings together three different mechanisms of development.

8.2 A transition mechanism theory for the ACT-R architecture

The preliminary implementation of the combinations of modifications shows that combined modifications enable the model to match the data better (on average) than individual modifications. The combined modifications can also be optimised to obtain the best possible match to the data. However, ACT-R is a general architecture and as such it has no theory for how developmental mechanisms interact with each other. The results of the combined modifications are therefore likely to match the seven year old children's data due to the sum of the modifications rather than their interaction. For example, how will a limit on conditions in productions affect an increase in expected gain noise (and vice versa)? The answer is unclear because not all of the modifications correspond to ACT-R mechanisms (i.e. some developmental mechanisms are implemented *within* the ACT-R

architecture, but are not *part of* the architecture). The architecture therefore provides no indication as to how the mechanisms will interact.

Specifying how the different developmental mechanisms (i.e. the modifications) interact would provide a more coherent model and is likely to provide a better match to the subject data. This section details a theory of maturation that could be implemented within the ACT-R cognitive architecture, which proposes some interactions between the mechanisms that influenced the behaviour of the Tower model in Chapter 7.

Three mechanisms will be considered in the theory of maturation within the ACT-R cognitive architecture: capacity, learning (i.e. knowledge), and processing speed. Processing speed will be put forward as the mediating factor in development. Speed in the ACT-R architecture provides absolute model timings which reduce the degrees of freedom when matching model–subject data. However, the transition mechanism theory will show how the timings can also be used to mediate capacity and learning.

ACT-R includes the time taken to match Declarative Memory Elements (DME's) in the timing for a production to fire. The firing latency for a production is the time taken to match each DME in the condition of the production, plus an additional set time to fire the rule. One way in which capacity was implemented in Chapter 7 was to limit the number of DME's that could be matched in the conditions of productions. This limit on the number of DME's that can be matched can be achieved automatically by using processing speed. By having a maximum time that a production can take to fire, a limit is placed on the number of DME's that can be matched in productions (because in general the time for a production to fire will correlate with the number of DME's in the condition of the production). Reducing the time that a production can take to fire therefore reduces the number of DME's that can be matched in productions. This provides a method by which processing speed mediates capacity.

Let us assume that through practice, the time to match a DME in the condition of a production decreases. This will enable more DME's to be matched in the conditions of productions and therefore can change the strategies that are used on a task. As the timing to match a DME decreases, productions can be matched which could not be matched before (because previously the time required to match all of the DME's was greater than the maximum time for a production to fire). This hypothesis would correspond to Case's

hypothesis (and possibly Halford's) that processing speed does not change, but through experience more elements can be simultaneously matched.

This account of capacity means that the number of conditions that can be matched in a production is actually dynamic. The amount varies based on the time allowed for a production to fire and how long each DME takes to be matched. It has been stated that "production-rule models can be made more realistic if we know how many conditions can be matched in each rule" (Halford, 1993, p.472). However, a dynamic account would seem more plausible: through experience, more conditions can be matched because of familiarity with the task.

Let us consider how new productions can be created. Experience with a task decreases the time required to match a DME. Assuming a fixed processing speed, sufficient experience will allow time for extra DME's, that are not part of the condition of the production, to be simultaneously matched. The most active DME'S at this point will be those that are most related to the task (this is part of the ACT-R theory). Consistencies can therefore be detected if the same DME's are simultaneously matched every time the production fires. These DME's can be added to the conditions of the production to make a more specific production. The production learning mechanism is mediated by processing speed because speed influences the number of DME's that can be matched in productions.

More general productions can be created by removing the most active DME's in the conditions of productions (i.e. the ones that require the least time to be matched). The most active DME's are the least likely to influence the firing of the production. The control of the production creation mechanism would itself take time and therefore would not be initiated until task experience means that there are processing resources to apply to it.

The hypothesis for transition mechanisms in ACT-R incorporates processing speed, capacity, and strategy acquisition. The proposed transition mechanism has omitted two developmental mechanisms that were shown in Chapter 7 to match the subject data well: strategy choice and strategy accuracy. Strategy choice can arise from selecting the production that matches its conditions the fastest. This makes encoding important because encoding the task correctly means the most appropriate DME's become the most active and therefore the appropriate productions (i.e. strategies) should be matched the fastest. Alternatively, the selection procedure from the ACT-R model presented in this

thesis could be used (based on success/failure flags). Strategy accuracy may directly result from the creation of more specific or more general productions, should they not lead to success.

The proposed transition mechanism is currently only a verbal hypothesis which will become more precise when it is implemented within an architecture. The transition mechanism includes components of ACT-R theory, Case's theory, and part of the transition mechanism implemented in GIPS. It follows from the basis of the thesis that the way forward in examining cognitive development is to examine what has been hypothesised in the past literature and take what appears to be the most plausible current account of events. The most plausible developmental mechanisms, based on the research that has been conducted here, have been selected and incorporated into a proposed transition mechanism.

It is possible that some of the methods that can be applied to the ACT-R architecture can also be applied to other rule-based architectures. Where the architecture represents productions and memory in a similar way to ACT-R the implementation may be straightforward. The more the architecture differs from ACT-R, the less the methods detailed above will apply.

8.3 Summary

The first part of this chapter showed the results of combinations of modifications. The combined modifications in general showed a better fit to the seven year old children's data than the individual modifications presented in Chapter 7, although the best of the individual modifications showed a slightly better fit to the subjects than the best of the combined modifications (because the combined modifications had not been optimised). The combined modifications showed that current and novel theories of development can be implemented and tested within a single model.

The second part of this chapter hypothesised how a theory of cognitive development could be implemented within the ACT-R cognitive architecture. This theory showed how the central mechanisms that influenced the behaviour of the model in Chapter 7 could interact in a coherent way. The theory provides the end-point of this thesis by bringing together previous and present research in a way which is both innovative and realisable. It continues the growing field of work which joins developmental theory and computational modelling in order to explain what develops.

9. Conclusions

The implications and conclusions of this work span four areas: how the work which has been done has helped examine what develops; the problems with the work presented; the useful contributions to knowledge that this work has provided; and the future work that the model can be used to examine. Each area is described, followed by an overall summary.

9.1 How this work has helped examine what develops

This thesis set out to show how computational modelling can help in examining development. Chapters 2 and 3 outlined various problems with theories and models of development: the verbal theories are vague and difficult to test; the models have failed to incorporate most theoretical mechanisms of development and match subject data on few measures. The modelling work presented here has shown that these problems can be overcome, and that models can provide specific benefits.

1. Testing mechanisms of development within a computational model showed which mechanisms influenced task behaviour, and how they influenced task behaviour.
2. Testing mechanisms of development has provided a focus for transition mechanisms because the mechanisms that influenced task behaviour have to be incorporated within any transition mechanism.
3. Model–subject comparisons were made on multiple measures, and this enabled a more detailed analysis of the influence that developmental mechanisms had on behaviour.
4. Learning within a task was explained.

Each of the above points will now be covered in more detail.

9.1.1 Developmental mechanisms have been tested

There is widespread belief that furthering our understanding of cognitive change is of critical importance to progressing developmental theory (e.g. Siegler, 1989; Sternberg, 1984). By investigating the role of different proposed mechanisms of development, this thesis has been able to shed light on the influence that each developmental mechanism has on behaviour (albeit in a single domain).

Chapter 3 showed that models of development have tended to concentrate on modifying task knowledge without concern for other mechanisms of development. Modifications

were made to the Tower model to represent a *range* of proposed mechanisms of development. No other computational model of development has examined a variety of developmental mechanisms. Mechanisms other than knowledge were shown to change the behaviour of the model such that the models performance began to look like the performance of seven year old children. This suggests that there are more mechanisms than knowledge that influence behaviour.

Testing theoretical mechanisms of development within a computational model also had a side-effect: any vague parts of the mechanisms became more detailed. The implementation of some mechanisms forced decisions to be made when the specification of the mechanism was imprecise. It could be argued that much of the added specification is because the mechanism is implemented within a computational framework and was the modeller's interpretation of what the theorists intention was. However, the implemented mechanisms still provide instances of precise versions of the theoretical mechanisms.

9.1.2 A focus for transition mechanisms has been provided

There are many computational models of development which match the subject data for each *discrete* level of performance, but they include no mechanisms by which transition *between* performance levels occurs. These models help the investigation of transition mechanisms by limiting the degrees of freedom for transition mechanisms (Simon & Halford, 1995). This is important because there are problems with current models which include transition mechanisms (see Chapter 3).

The research here helps the investigation of transition mechanisms by providing a focus to guide their development. Chapter 2 described many developmental mechanisms, most of which had not been previously examined within a computational model. The majority of the mechanisms examined significantly altered the behaviour of the Tower model. By eliciting the range of developmental mechanisms that influence performance, a checklist of behaviour is obtained which a transition mechanism must be able to account for. For example, increasing capacity must improve behaviour; having more accurate strategies must improve behaviour.

The transition mechanism cannot simply be a *learning* mechanism, because the modifications showed that developmental mechanisms interact with learning mechanisms. Implementing some mechanisms meant the learning in the model was disrupted. The

mechanism cannot simply be a *developmental* mechanism, because this will not account for within-task learning.

Klahr (1995) argues that computational models must always be undergoing self-modification; the stage-then-transition description of development presented by Simon (1962) is too simple. Development is therefore viewed as a self-modifying system. The scope of the self-modifying system is not detailed. It is unclear whether Klahr includes both learning mechanisms and developmental mechanisms within a self-modifying system.

The Tower model is consistent with Klahr's view, and is able to expand upon it. The model shows that as learning mechanisms interact with developmental mechanisms, a self-modifying system has to include both learning *and* developmental mechanisms. This means that learning and development are not strictly separate entities, although this approach is consistent with current connectionist models of development (e.g. Shultz, Schmidt, Buckingham & Mareschal, 1995) where development occurs because of learning.

If the behaviour of the system, or model, is continually under revision (as a self-modifying system has to be), it suggests that development is not simply stage-like. In tasks like the balance-scale, children's performance can be separated into stages whilst models of the task, such as the McClelland and Jenkins model, do not perform in a stage-like manner (Raijmakers, Van Koten & Molenaar, 1996). However, it is not clear that performance which conforms to stagewise development is important. To researchers involved in modelling development, the importance lies in modelling the behaviour at different performance levels, and in being able to represent the transition between general performance levels.

9.1.3 Detailed model–subject matching can be done

Previous computational models of development were compared with subject data on very few measures. This is because of the task they modelled, the behaviour that empirical studies have measured, or the measures that the model can output. The Tower model showed that model–subject data can and should be compared using a variety of measures. Comparing model and subject data on a variety of measures provided a more detailed match to the subject data, and provided more faith that the processes modelled were correct.

Matching subject behaviour on a range of measures enabled a detailed examination of the influence each developmental mechanism had on behaviour. For example, reducing the number of objects that could be attended to increased the visual search time. This example also shows the importance of using measures that describe behaviour at a low-level. The increase in visual search time would never have been noticed if only high-level measures had been taken. It is therefore important to examine as many measures as possible, and in as much detail as possible, so that the processes used by the model can be justified and improved, and each different mechanism can be qualitatively assessed.

One problem of attempting to fit the subject data on multiple measures is that it is very difficult to provide a match on every measure. The original model matched the data from adults on 7 out of 9 measures, and the modified models fit the data from seven year old children on 7 out of 9 measures at best. A fit on every measure was not essential to achieve the goal of this thesis, because the influence of modifications can be seen in the *changes* in behaviour. Providing a fit on every measure may not be necessary for many other tasks. In any case, the measures which the model does not fit the subject data on provide the areas where the model needs to be improved.

9.1.4 Learning within tasks can be modelled and explicated

The Tower task included within-task learning. Computational models are seen to be the only way in which learning on a task can be observed directly (Simon & Halford, 1995). This was the case for the Tower task: the reduced times for subsequent layers that subjects construct could have arisen because of the reduced visual search that is required. The model of the task explained where the time was spent for each layer. This revealed that a lot of learning occurred when the first two layers were constructed (for the original model).

The model also enabled a detailed view of *what* was learned. The fluctuations in the expected gain values of productions and the productions created from learning could be seen. The time course of the model showed that the general block fitting productions that the model began with declined in their expected gain values, and deciding how to fit blocks became governed by the newly created productions.

The modifications affected the learning mechanisms in different ways. Many of the knowledge modifications had no effect on behaviour because they were compensated for

by the learning within the model. The strategy choice modification slowed down the learning mechanism and enabled the model's timings in completing each layer to match those of seven year olds. The modifications have clearly showed that there is an interaction between developmental mechanisms and learning mechanisms.

9.2 Problems with the work presented

Throughout this thesis various problems have been outlined concerning computational modelling, and with the way in which computational modelling was used in the thesis. Some of these problems will be highlighted and expanded upon here.

9.2.1 Model predictions were not tested against new empirical data

The model presented in this thesis represents a theory of how the Tower task is accomplished. A drawback of the thesis is that predictions from the model were not tested against new data on the task. This would have provided strength for the belief that the model completes the task in the same way that subjects do. Various task predictions can be made, such as:

- Sorting the blocks so that they are grouped by size when on the table should make the task a lot easier because the need for visual search is reduced. The model predicts that the majority of the reduction in task time would be because less searching for blocks of the same size is necessary. This prediction is currently difficult to test in subjects because of technical difficulties in examining eye movements whilst completing the task.
- Removing different features of the blocks should make the task easier, because less features have to be attended to. Furthermore, the removal of some features may make the task easier than when removing other features.
- Highlighting certain features may make the task easier or more difficult to complete, depending whether the highlighted features are task appropriate or not.

These are the types of predictions for which new task data could be obtained. The simulation that the model is linked to could be altered to reflect these task changes (for the first prediction, only the initial layout of blocks need change), and the new empirical data could be tested against the predictions made by the model. This type of work is a natural extension to this thesis because there must be confidence that the model is performing the task in a similar fashion to subjects. When this is the case, the impact of implemented developmental mechanisms on the model's behaviour can be taken seriously.

9.2.2 The developmental mechanisms were not tested in other domains

The developmental mechanisms were examined within a domain-specific environment. It has been argued that the task for developmental researchers "is to identify the[se] mechanisms [of development] with greater precision, to determine which are general and which are domain specific, and to discover how both types of mechanism interact in the course of development" (Kail & Bisanz, 1992, p.244). Although the theories from which the mechanisms have been taken are all domain general (suggesting that the mechanisms should apply to other domains), the opportunity to test this has not been taken, and presents a drawback for this thesis.

Testing developmental mechanisms within the same computational framework but across domains will determine which mechanisms are domain-specific and which are domain-general. The thesis has shown that implementing the mechanisms is possible; further work must therefore examine to what extent the developmental mechanisms influence behaviour in other tasks.

9.2.3 Starting with a model of adult behaviour

Trying to work backwards from a model of adult behaviour has been openly criticised (Klahr, 1984b; 1995). Modelling adult behaviour and trying to regress from it (to children's behaviour) may well mask a lot of the underlying structure that children may require (Norman, 1980). For example, adults may have automatic processes that children do not.

The work presented here investigated the influence of developmental mechanisms, rather than attempting to suggest the exact processes that the children used to solve the Tower task. The central aim was to show how each developmental mechanism affected the behaviour of the model, rather than show what changes cause the model to match the data from seven year old's. Regressing from adult behaviour is a problem, but its effect was limited within the scope of this thesis.

9.2.4 The model fails to capture all of the subject behaviour

All computational models fail to capture some aspects of subject behaviour. There are usually various reasons for this: the behaviour is not important to the task; the behaviour is not important to the research questions being asked; time limitations mean that some measures had to be discarded. For the Tower task, some subject behaviour was not

measured, and some aspects of the task simulation were not ideal matches to human physiology:

1. There were measures of how blocks were manipulated, and how blocks were selected, that were omitted from the analyses in Chapter 4. This was mainly because the measures were subjective. With more time and labour these measures could be taken reliably, and would thereby provide predictions that the Tower model could be tested against.
2. The model and task simulation failed to perform motor actions in parallel. For example, one hand could not be moved to grasp a block at the same time as the other hand was moved to grasp a different block. This should have meant the model performed the task slower than adults. However, the timing estimate of 550 ms per hand movement was generous in order to average the time for parallel and serial hand movements.
3. The viewing direction of the eye was not taken into account when examining blocks. This meant that if the block had a hole on the opposite side to where the eye was, the hole was still seen. This tended to only be a factor for the first layer, when the characteristics of blocks were unknown.
4. The simulation hands did not pass any haptic information. The lack of haptic information goes some way to cancelling out ignoring the viewing direction of the eye, because often the hands can feel what features the block has. This would have reduced the need for the model to rotate a block to see its features.

9.2.5 Are there other unexplored developmental mechanisms?

The mechanisms that were examined within this thesis are all mechanisms that have been put forward by theories of development. There are further mechanisms that are implied by studies of development. For example, encoding and analogy were explained in previous chapters (see also Siegler, 1989).

When a problem has a large number of elements, only some will be encoded and used toward finding a solution to the problem. If a solution was not found, other problem elements may be encoded. Encoding usually entails the most salient problem elements to be encoded first, unless experience with the type of problem suggest otherwise (for example, Chapter 3 suggested that young children on the balance-scale task encode weight and older children encode both weight and distance).

Analogy can be used when a child is confronted with a problem situation that has not been seen before, but the child does have experience in other similar problems. The experience with the other problems can be used, via analogy, to help with the current problem situation. One method of analogy, based on relational similarity, was explored by Gentner, Rattermann, Markman and Kotovsky (1995) and was detailed in Chapter 3.

The mechanisms of analogy and encoding have not been explored *in detail* within any computational model of development. Encoding could be a factor in the Tower task because of the variety of features that each block has. The Tower model could therefore provide an ideal vehicle for testing the encoding mechanism.

9.3 Contributions of this work

The work detailed in this thesis represents several contributions to the knowledge and study of development, modelling, learning, and interaction: using multiple measures provides a better match to the subject data; modifying models provides a new methodology for testing models; a new way of representing capacity has been provided; methods for analysing where the task time goes have been provided; tools to aid modelling have been developed. These contributions will be detailed in turn.

9.3.1 Using multiple measures provide a better match to subject data

The model uses multiple measures to match its behaviour with subject behaviour. The full range of measures used gives a good impression of which areas the model provides a good fit to the data, and which areas the model could be improved. Previous models of developmental tasks have tended to match subject behaviour on one or only a small number of measures (e.g. percentage of correct responses). This reduces the opportunity to discover areas where the model needs improving.

The timing estimates that are included in the model mean that matches between the model's behaviour and the subject's behaviour is at a low-level of detail. This adds a degree of confidence that the model is using the correct types of process to match the subject data, because the model is able to match the timings of subjects at different stages of completion of the task. The timing estimates can also help to identify where learning is occurring during the task. All of the previous models of development have ignored timing estimates. Ignoring timing estimates leaves a model more open to criticism, and may mean that learning on the task is difficult to identify. In some respects this is task dependent,

but all of the models described in Chapter 3 used tasks for which timing data would have been simple to obtain.

9.3.2 Modifying models provides a new methodology for testing models

The modifications have highlighted some knowledge that is redundant within the model, for its removal makes no difference to the behaviour of the model (for example, removing the knowledge that pegs fit into holes, see Chapter 7). This presents an interesting method by which knowledge and processes in models can be tested to see whether they do influence the behaviour of the model. In addition, it shows the necessary knowledge for the model to match the subject behaviour (for example, Chapter 7 showed that removing the internal obstructing feature check improved the behaviour of the model). Modifying the model has therefore had the side-effect of producing a method by which the basic knowledge and processes included in a computational model can be tested to see the actual effect they have on the model's behaviour.

The idea of testing a model by modifying its contents was first presented by Simon and Halford (1995) as a method of finding the best model that matches the data when there are different processing and representation options available. The work here has shown that this idea can be extended: modelling can help ascertain which processes and representations are *necessary* in completing the task, as well as differentiating competing processes and representations.

9.3.3 A new way of representing capacity in rule based architectures

Two ways of representing capacity were introduced. The first was the most obvious way of viewing capacity in rule-based models. This way subjected memory elements to decay, and ignored such elements if they fell below a particular threshold (this method was used to operationalise Pascual-Leone's M-power). However, the view of capacity as being a set number of items that can be held in memory tends to be seen as outdated, because as the number of items to process increases, performance declines but does not cease (Simon & Halford, 1995). The connectionist view of capacity fits this concept because performance in connectionist networks declines with limited capacity, but does not cease altogether. It is possible that the traditional rule based decay mechanism could result in all elements being below threshold and so behaviour halts.

The second way of representing capacity in the Tower model was limiting the number of conditions that production rules had. The limit on the number of conditions showed the

kind of performance that connectionist networks show. Performance significantly declined because as capacity became more limited, the number of productions required to perform the task increased. The reduction in performance was demonstrated in the Case and Halford modifications in Chapter 7. The new method of operationalising capacity in rule based models means that the examination of capacity need not be restricted to connectionist based models only.

9.3.4 Methods for analysing where the task time goes have been provided

Including timing estimates and a simulation of the task enabled a detailed account of the time taken to complete the Tower. The time taken could be split into the time spent processing (cognition time), searching (eye time), and manipulating blocks (hand time). The analysis of time taken revealed that for the original model, only around 50% of the task time was taken up by cognition. This shows that any model of a task which involves interaction with an environment will overestimate process time unless a task simulation is included.

9.3.5 Tools to aid modelling have been developed

This work was greatly helped by writing functions within the model so that the behavioural data was output in the same format as the coded transcriptions of adult and seven year old behaviour. Once both the subject and model data are in exactly the same formats, analysis is made much simpler.

However, the analysis of the model's behaviour could have been made much simpler if ACT-R included tools for understanding the model. For example, obtaining analyses on how often a specific production fired required independent functions to be written. Full analyses of the model's behaviour therefore incorporated many such functions. This now enables examination of the performance of many ACT-R models to be simpler. The functions which enable the detailed analysis of ACT-R models have now been written as part of this thesis.

9.4 Future work

The thesis has already outlined some further work that needs to be done (in terms of detailing some problems with the thesis), in section 9.2. In addition, section 8.2 outlined a transition mechanism that can be implemented within the ACT-R architecture. The theory is currently expressed in verbal terms only. With further work, the hypothesised mechanism can be realised as an actual transition mechanism. There are also extensions

that can be made to the model to enable related domains to be explored. These are outlined below.

Tutoring strategies can be implemented in the model so that each tutoring strategy can be defined more precisely. First, the model can suggest how much and what type of tutoring is sufficient to complete the task. Second, the model can suggest the kind of knowledge that each tutoring strategy gives. The model can therefore test claims of how tutoring strategies increase the knowledge base. This was not within the scope of this thesis, but developing the model to include tutoring appears to be within reach. For example, some types of tutoring may elicit better task strategies. In the model, the productions representing these strategies can be given higher expected gain values.

The model can also determine when mental models are used when completing the Tower task. The mental model incorporated into the Tower model fits together two blocks internally rather than physically carrying out the construction. The model already suggests that subjects either have a noisy internal fitting procedure, or sometimes they do not carry out the procedure, because they still produce erroneous constructions. The precise use of mental models is not explored in this thesis, but some questions could be answered with some further work to the model. When should an internal fit be done, and how noisy should the results from it be? Is the internal fit procedure only used in situations when it is easy to provide a mental image of the internal fit? Are some individuals more inclined to attempt a physical fit than use mental processing as an internal fitting check? Why should this be the case? Does it make their performance better? Is it partly due to a speed/accuracy trade-off?

The easiest work for future implementation is the work on instruction and tutoring because the Tower task was designed to investigate these areas. Including tutoring strategies within the model is not a large step away, and the subject data regarding tutoring strategies is already in hand. However, the most innovative and perhaps the most fruitful is the implementation of a transition mechanism within ACT-R. The developmental community lacks an architecture (outside of connectionism) in which to model developmental phenomena, because the architectures are not intended for modelling developmental phenomena. If ACT-R can include a transition mechanism then an architecture additional to connectionism is provided for researchers who are using modelling to examine what develops.

9.5 Summary

Theories of development have proposed different mechanisms of development, which have not previously been rigorously tested. The thesis has tested the mechanisms and has shown which influence behaviour and which do not. The work is a novel method of examining child development and is a step forward in examining both cognitive development in general, and the transition mechanisms that enable development.

Acknowledgements

Frank Ritter for providing all of the help and support that a postgraduate could need, and for reading all previous drafts of this thesis.
David Wood and Richard Young for various discussions on development and modelling, and for comments on thesis chapters.
Pete Bibby, Julian Pine, and Mike Byrne for comments on thesis chapters.
Heather Wood for providing the videos of seven and five year old children completing the Tower task.

References

Anderson, J. R. (1993). *Rules of the mind.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought.* Mahwah, NJ: Lawrence Erlbaum Associates.

Anderson, J. R., Matessa, M., & Lebiere, C. (1997). ACT-R: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction, 12,* 439-462.

Andrews, S. (1989). Frequency and neighbourhood effects on lexical access: Activation or search? *Journal of Experimental Psychology: Learning, Memory and Cognition, 15,* 802-814.

Atkinson, R. L., Atkinson, R. C., Smith, E. E., Bem, D. J., & Hilgard, E. R. (1990). *Introduction to Psychology, Tenth Edition.* Orlando, Florida: Harcourt Brace Jovanovich.

Baxter, G. D., & Ritter, F. E. (1996). *Designing abstract visual perceptual and motor action capabilities for use by cognitive models.* (Technical Report 36). CREDIT, Psychology Department: University of Nottingham.

Bechtel, W. (1993). The case for connectionism. *Philosophical Studies, 71,* 119-154.

Bechtel, W., & Abrahamsen, A. (1991). *Connectionism and the mind: An introduction to parallel processing in networks.* Oxford, UK: Blackwell.

Beilin, H. (1992). Piagetian theory. In R. Vasta (Ed.), *Six theories of child development: Revised formulations and current issues.* London and Philadelphia: Jessica Kingsley.

Beilin, H. (1994). Mechanisms in the explanation of developmental change. In H. W. Reese (Ed.), *Advances in child development and behavior,* (Vol. 25). London: Academic Press.

Broadbent, D. E. (1970). Psychological aspects of short-term and long-term memory. In *Proceedings of the Royal Society,* 333-350.

Brown, A., & DeLoache, J. S. (1978). Skills, plans, and self-regulation. In R. Siegler (Ed.), *Children's thinking: What develops?,* 3-35. Hillsdale, NJ: Lawrence Erlbaum Associates.

Buckingham, D., & Shultz, T. R. (1994). A connectionist model of the development of velocity, time, and distance concepts. In *Proceedings of the Sixteenth Annual Conference of the Cognitive Society Conference,* 72-77. Hillsdale, NJ: Lawrence Erlbaum Associates.

Byrne, M. D., & Anderson, J. R. (1997). Enhancing ACT-R's perceptual-motor abilities. In *Proceedings of the Nineteenth Cognitive Science Conference*. Stanford, CA: Lawrence Erlbaum Associates.

Card, S. K., Moran, T. P., & Newell, A. (1983). *The psychology of human-computer interaction*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Carpenter, R. H. S. (1977). *Movements of the eyes*. London: Pion.

Case, R. (1978). Intellectual development from birth to adulthood: A neo-Piagetian interpretation. In R. S. Siegler (Ed.), *Children's thinking: What develops?* Hillsdale, NJ: Lawrence Erlbaum Associates.

Case, R. (1985). *Intellectual development: Birth to adulthood*. Orlando, Florida: Academic Press.

Donaldson, M. (1978). *Children's minds*. London: Fontana Press.

Elkind, D. (1961). Children's discovery of the conservation of mass, weight, and volume: Piagetian replication study II. *Journal of Genetic Psychology, 98,* 219-227.

Elman, J. L., Bates, E. A., Johnson, M. H., Karmiloff-Smith, A., Parisi, D., & Plunkett, K. (1996). *Rethinking innateness: A connectionist perspective on development*. Cambridge, MA: MIT Press.

Ericsson, K. A., & Simon, H. A. (1984). *Protocol analysis: Verbal reports as data*. Cambridge, MA: MIT Press.

Fahlman, S. E., & Lebiere, C. (1990). The cascade-correlation learning architecture. In D. S. Touretzky (Ed.), *Advances in neural information processing systems 2*, 524-532. Los Altos, CA: Morgan Kaufmann.

Fisher, D. H., Pazzani, M. J., & Langley, P. (1991). *Concept formation: Knowledge and experience in unsupervised learning*. Los Altos, CA: Morgan Kaufmann.

Flavell, J. H. (1978). Comments. In R. S. Siegler (Ed.), *Children's thinking: What develops?* Hillsdale, NJ: Lawrence Erlbaum Associates.

Flavell, J. H. (1984). Discussion. In R. J. Sternberg (Ed.), *Mechanisms of cognitive development*. Prospect Heights, Illinois: Waveland Press.

Flavell, J. H. (1996). Piaget's legacy. *Psychological Science, 7,* 200-203.

Flavell, J. H., Miller, P. H., & Miller, S. (1993). *Cognitive development*. (3rd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Flavell, J. H., & Wellman, H. M. (1977). Metamemory. In R. V. Kail & J. W. Hagen (Eds.), *Perspectives on the development of memory and cognition*, 3-33. Hillsdale, NJ: Lawrence Erlbaum Associates.

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

Fry, A. F., & Hale, S. (1996). Processing speed, working memory, and fluid intelligence. *Psychological Science, 7,* 237-241.

Gelman, R. (1972). The nature and development of early number concepts. In H. W. Reese (Ed.), *Advances in child development and behavior,* (Vol. 7). London: Academic Press.

Gelman, R. (1982). Accessing one-to-one correspondence: Still another paper about conservation. *British Journal of Psychology, 73,* 209-220.

Gentner, D., Rattermann, M. J., Markman, A., & Kotovsky, L. (1995). Two forces in the development of relational similarity. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling,* 263-313. Hillsdale, NJ: Lawrence Erlbaum Associates.

Gobet, F. (in press). Simulations of stagewise development with a symbolic architecture. In W. Tschacher & J. Dauwalder (Eds.), *Dynamics, synergetics, autonomous agents - Nonlinear systems approaches to cognitive psychology and cognitive science.* Singapore: World Scientific.

Grainger, J., & Segui, J. (1990). Neighbourhood frequency effects in visual word recognition: A comparison of lexical decision and masked identification latencies. *Perception & Psychophysics, 47,* 191-198.

Grant, D. A. (1962). Testing the null hypothesis and the strategy and tactics of investigating theoretical models. *Psychological Review, 69,* 54-61.

Halford, G. S. (1989). Reflections on 25 years of Piagetian cognitive developmental psychology, 1963-1988. *Human Development, 32,* 325-387.

Halford, G. S. (1993). *Children's understanding: The development of mental models.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Halford, G. S. (1995). Learning processes in cognitive development: A reassessment with some unexpected implications. *Human Development, 38,* 295-301.

Halford, G. S., Maybery, M. T., & Bain, J. D. (1986). Capacity limitations in children's reasoning: A dual-task approach. *Child Development, 57,* 616-627.

Halford, G. S., Smith, S. B., Campbell-Dickson, J., Maybery, M. T., Kelly, M. E., Bain, J. D., & Stewart, J. E. M. (1995). Modeling the development of reasoning strategies: The role of analogy, knowledge, and capacity. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling,* 77-156. Hillsdale, NJ: Lawrence Erlbaum Associates.

Inhelder, B., & Piaget, J. (1958). *The growth of logical thinking from childhood to adolescence.* London: Routledge & Kegan Paul.

Jensen, E. M., Reese, E. P., & Reese, T. W. (1950). The subitizing and counting of visually presenting fields of dots. *Journal of Psychology, 30,* 363-392.

Jones, G. (1996b). The architectures of Soar and ACT-R, and how they model human behaviour. *AISB Quarterly, Winter 1996,* 41-44.

Jones, R. M., & VanLehn, K. (1991). Strategy shifts without impasses: A computational model of the sum-to-min transition. In *Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 358-363. Hillsdale, NJ: Lawrence Erlbaum Associates.

Kail, R. (1986). Sources of age differences in speed of processing. *Child Development, 57,* 969-987.

Kail, R. (1988). Developmental functions for speeds of cognitive processes. *Journal of Experimental Child Psychology, 45,* 339-364.

Kail, R. (1991). Development of processing speed in childhood and adolescence. In H. W. Reese (Ed.), *Advances in child development and behavior*, (Vol. 23), 151-185. London: Academic Press.

Kail, R. (1996). Nature and consequences of developmental change in speed of processing. *Swiss Journal of Psychology, 55,* 133-138.

Kail, R., & Bisanz, J. (1992). The information-processing perspective on cognitive development in childhood and adolescence. In R. J. Sternberg & C. A. Berg (Eds.), *Intellectual development,* 229-260. Cambridge: Cambridge University Press.

Karmiloff-Smith, A., & Inhelder, B. (1974). If you want to get ahead, get a theory. *Cognition, 3,* 195-212.

Kennedy, R. A., & Murray, W. S. (1986). The components of reading time: Eye movement patterns of good and poor readers. In *Proceedings of the Third European Conference on Eye Movements.* Amsterdam: Elsevier.

Kieras, D. E., & Meyer, D. E. (1996). The EPIC architecture for human cognition and performance modeling. In *Proceedings of the Third Annual ACT-R Workshop.*

Klahr, D. (1984a). Transition processes in quantitative development. In R. Sternberg (Ed.), *Mechanisms of cognitive development,* 101-139. San Francisco: W. H. Freeman.

Klahr, D. (1984b). Commentary: An embarrassment of number. In C. Sophian (Ed.), *Origins of cognitive skills,* 295-309. Hillsdale, NJ: Lawrence Erlbaum Associates.

Klahr, D. (1992). Information-Processing Approaches. In R. Vasta (Ed.), *Six theories of child development: Revised formulations and current issues,* 133-185. London and Philadelphia: Jessica Kingsley.

Klahr, D. (1995). Computational models of cognitive change: State of the art. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling*, 355-375. Hillsdale, NJ: Lawrence Erlbaum Associates.

Klahr, D., & Siegler, R. S. (1978). The representation of children's knowledge. In H. W. Reese & L. P. Lipsett (Eds.), *Advances in child development and behavior*, (Vol. 12). New York: Academic Press.

Klahr, D., & Wallace, J. G. (1976). *Cognitive development: An information processing view*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). Soar: An architecture for general intelligence. *Artificial Intelligence, 33,* 1-64.

Langley, P. (1987). A general theory of discrimination learning. In D. Klahr, P. Langley, & R. Neches (Eds.), *Production system models of learning and development*. Cambridge, MA: MIT Press.

Lebiere, C., Wallach, D., & Taatgen, N. (1998). Implicit and explicit learning in ACT-R. In *Proceedings of the Second European Conference on Cognitive Modelling*, 183-189. Nottingham, UK: Nottingham University Press.

Lemaire, P., & Siegler, R. S. (1995). Four aspects of strategic change: Contributions to children's learning of multiplication. *Journal of Experimental Psychology: General, 124,* 83-97.

Lovett, M. C., Reder, L. M., & Lebiere, C. (1997). Modeling individual differences in a digit working memory task. In *Proceedings of the Nineteenth Annual Conference of the Cognitive Science Society*, 460-465. Mahwah, NJ: Lawrence Erlbaum Associates.

Mareschal, D., Plunkett, K., & Harris, P. (1995). Developing object permanence: A connectionist model. In *Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society*, 170-175. Mahwah, NJ: Lawrence Erlbaum Associates.

Matessa, M., & Anderson, J. R. (1996). An ACT-R model of menu search. In *Proceedings of the Third Annual ACT-R Workshop*.

McClelland, J. L., & Jenkins, E. (1991). Nature, nurture and connections: Implications of connectionist models for cognitive development. In K. VanLehn (Ed.), *Architectures for intelligence*. Hillsdale, NJ: Lawrence Erlbaum Associates.

McClelland, J. L., & Rumelhart, D. E. (1981). An interactive activation model of context effects in letter perception. Part 1, An account of basic findings. *Psychological Review, 88,* 375-407.

McCloskey, M. (1991). Networks and theories: The place of connectionism in cognitive science. *Psychological Science, 2,* 387-395.

McLaughlin, B. P. (1993). The connectionism/classicism battle to win souls. *Philosophical Studies, 71,* 163-190.

Meyer, D. E., & Kieras, D. E. (1997). A computational theory of executive cognitive processes and multiple-task performance: Part 1. Basic mechanisms. *Psychological Review, 104,* 3-65.

Miller, P. H. (1993). *Theories of developmental psychology.* (3rd ed.). New York: W. H. Freeman and Company.

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

Murphy, C. M., & Wood, D. J. (1981). Learning from pictures: The use of pictorial information by young children. *Journal of Experimental Child Psychology, 32,* 279-297.

Murphy, C. M., & Wood, D. J. (1982). Learning through media: A comparison of 4-8 year old children's responses to filmed and pictorial instruction. *International Journal of Behavioural Development, 5,* 195-216.

Murray, F. B. (1991). Questions a satisfying developmental theory would answer: The scope of a complete explanation of developmental phenomena. In H. W. Reese (Ed.), *Advances in child development and behavior,* (Vol. 23). London: Academic Press.

Myers, B. A., Guise, D. A., Dannenberg, R. B., Vander Zanden, V., Kosbie, D. S., Pervin, E., Mickish, A., & Marchal, P. (1990). Garnet: Comprehensive support for graphical, highly-interactive user interfaces. *IEEE Computer, 23,* 71-85.

Nelson, G., Lehman, J. F., & John, B. E. (1994). Integrating cognitive capabilities in a real-time task. In *Proceedings of the Proceedings of the Sixteenth Annual Conference of the Cognitive Science Society.*

Nerb, J., Krems, J. F., & Ritter, F. E. (1993). Rule learning and the power law: A computational model and empirical results. In *Proceedings of the Fifteenth Annual Conference of the Cognitive Science Society,* 765-770. Hillsdale, NJ: Lawrence Erlbaum Associates.

Newell, A. (1968). On the analysis of human problem solving protocols. In J. C. Gardin & B. Javlin (Eds.), *Calcul et formalization dan les sciences de l'homme.* Paris: Center de la Researche Scientifique.

Newell, A. (1972). A note on process-structure distinctions in developmental psychology. In S. Farnham-Diggory (Ed.), *Information processing in children.* New York: Academic Press.

Newell, A. (1990). *Unified theories of cognition.* Harvard: Harvard University Press.

Newell, A., & Rosenbloom, P. S. (1981). Mechanisms of skill acquisition and the power law of practice. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*, 1-55. Hillsdale, NJ: Lawrence Erlbaum Associates.

Norman, D. A. (1980). Twelve issues for cognitive science. *Cognitive Science, 4,* 1-32.

Ohlsson, S., & Langley, P. (1986). *PRISM tutorial and manual.* Irvine: University of California Press.

Pascual-Leone, J. (1969). *Cognitive development and cognitive style.* Unpublished doctoral dissertation, University of Geneva.

Pascual-Leone, J. (1970). A mathematical model for the transition rule in Piaget's developmental stages. *Acta Psychologica, 32,* 301-345.

Pascual-Leone, J. (1987). Organismic processes for neo-Piagetian theories: A dialectical causal account of cognitive development. In A. Demetriou (Ed.), *The neo-Piagetian theories of cognitive development: Towards an integration.* Amsterdam: North-Holland.

Pascual-Leone, J. (1995). Learning and development as dialectical factors in cognitive growth. *Human Development, 38,* 338-348.

Pascual-Leone, J., & Morra, S. (1991). Horizontality of water-level: A neo-Piagetian developmental review. In H. W. Reese (Ed.), *Advances in child development and behavior,* (Vol. 23). London: Academic Press.

Piaget, J. (1950). *The psychology of intelligence.* London: Routledge & Kegan Paul.

Piaget, J. (1952). *The origins of intelligence in children.* New York: International Universities Press.

Piaget, J. (1960). The general problems of the psychobiological development of the child. In J. M. Tanner & B. Inhelder (Eds.), *Discussions on child development,* (Vol. 4). New York: International Universities Press.

Plunkett, K., & Marchman, V. (1993). From rote learning to system building: Acquiring verb morphology in children and connectionist nets. *Cognition, 48,* 21-69.

Plunkett, K., & Sinha, C. (1992). Connectionism and developmental theory. *British Journal of Developmental Psychology, 10,* 209-254.

Raijmakers, M. E. J., Van Koten, S., & Molenaar, P. C. M. (1996). On the validity of simulating stagewise development by means of PDP networks: Application of catastrophe analysis and an experimental test of rule-like network performance. *Cognitive Science, 20,* 101-136.

Reichgelt, H., Shadbolt, N. R., Paskiewicz, T., Wood, D. J., & Wood, H. (1993). EXPLAIN: On implementing more effective tutoring systems. In A. Sloman, D. Hogg,

G. Humphreys, D. Partridge, & A. Ramsay (Eds.), *Prospects for Artificial Intelligence*, 239-249. Amsterdam: IOS Press.

Rieman, J., Lewis, C., Johnson, Young, R. M., & Polson, P. G. (1994). "Why is a raven like a writing desk?": Lessons in interface consistency and analogical reasoning from two cognitive architectures. In *Proceedings of the Proceedings of CHI '94: Human Factors in Computing Systems*, 438-444. ACM Press.

Ritter, F. E., & Larkin, J. H. (1994). Using process models to summarize sequences of human actions. *Human-Computer Interaction, 9,* 345-383.

Rosser, R. (1994). *Cognitive development: Psychological and biological perspectives.* Boston: Allyn and Bacon.

Rumelhart, D. E., McClelland, J. L., & Group, t. P. R. (1986). *Parallel Distributed Processing: Explorations in the microstructure of cognition.* Cambridge, MA: MIT Press.

Searle, J. R. (1980/1997). Minds, brains, and programs. In J. Haugeland (Ed.), *Mind Design II.* Cambridge, Mass: MIT Press.

Shultz, T. R. (1992). Choosing a unifying theory for cognitive development. *Behavioral and Brain Sciences, 15,* 456-457.

Shultz, T. R., Mareschal, D., & Schmidt, W. C. (1994). Modeling cognitive development on balance scale phenomena. *Machine Learning, 16,* 57-86.

Shultz, T. R., Schmidt, W. C., Buckingham, D., & Mareschal, D. (1995). Modeling cognitive development with a generative connectionist algorithm. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling*, 205-261. Hillsdale, NJ: Lawrence Erlbaum Associates.

Siegler, R. S. (1976). Three aspects of cognitive development. *Cognitive Psychology, 8,* 481-520.

Siegler, R. S. (1981). Developmental sequences within and between concepts. *Monographs of the Society for Research in Child Development, 46,* 1-74.

Siegler, R. S. (1987). The perils of averaging data over strategies: An example from children's addition. *Journal of Experimental Psychology: General, 116,* 250-264.

Siegler, R. S. (1989). Mechanisms of cognitive development. *Annual Review of Psychology, 40,* 353-379.

Siegler, R. S. (1991). *Children's thinking.* (2nd ed.). Englewood Cliffs, NJ: Prentice-Hall.

Siegler, R. S. (1995). Children's thinking: How does change occur? In W. Schneider & F. Weinert (Eds.), *Memory, performance and competencies: Issues in growth and development.*, 405-430. Hillsdale, NJ: Lawrence Erlbaum Associates.

Siegler, R. S., & Jenkins, E. A. (1989). *How children discover new strategies*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Siegler, R. S., & Shipley, C. (1995). Variation, selection and cognitive change. In T. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling*. Hillsdale, NJ: Lawrence Erlbaum Associates.

Siegler, R. S., & Shrager, J. (1984). Strategy choices in addition and subtraction: How do children know what to do? In C. Sophian (Ed.), *Origins of cognitive skills*, 229-293. Hillsdale, NJ: Lawrence Erlbaum Associates.

Simon, H. A. (1962). An information processing theory of intellectual development. In W. Kessen & C. Kuhlman (Eds.), *Thought in the young child*, 127-131. Yellow Springs, OH: The Antioch Press.

Simon, H. A. (1972). On the development of the processor. In S. Farnham-Diggory (Ed.), *Information processing in children*. New York: Academic Press.

Simon, T., & Klahr, D. (1995). A computational theory of children's learning about number conservation. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling*, 315-353. Hillsdale, NJ: Lawrence Erlbaum Associates.

Simon, T., Newell, A., & Klahr, D. (1991). A computational account of children's learning about number conservation. In P. S. Rosenbloom, J. E. Laird, & A. Newell (Eds.), *The Soar Papers*, (Vol. Two), 1360-1399. Cambridge, Mass: MIT Press.

Simon, T. J., & Halford, G. S. (1995). Computational models and cognitive change. In T. J. Simon & G. S. Halford (Eds.), *Developing cognitive competence: New approaches to process modeling*, 1-30. Hillsdale, NJ: Lawrence Erlbaum Associates.

Sperling, G. A. (1960). The information available in brief visual presentation. *Psychological Monographs, 74.*

Sternberg, R. J. (1984). Mechanisms of cognitive development: A componential approach. In R. J. Sternberg (Ed.), *Mechanisms of cognitive development*, 163-186. Prospect Heights, IL: Waveland Press.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem solving strategies. *Cognitive Science, 15.*

Vygotsky, L. S. (1978). *Mind in society*. Cambridge, NY: Cambridge University Press.

Vygotsky, L. S. (1981). The genesis of higher mental functions. In J. V. Wertsch (Ed.), *The concept of activity in Soviet psychology*. Armonk, NY: Sharpe.

Wertsch, J. V., & Kanner, B. G. (1992). A sociocultural approach to intellectual development. In R. L. Sternberg & C. A. Berg (Eds.), *Intellectual development*. Cambridge, NY: Cambridge University Press.

Wilkening, F. (1981). Integrating velocity, time, and distance information: A developmental study. *Cognitive Psychology, 13,* 231-247.

Wilkening, F. (1982). Children's knowledge about time, distance, and velocity interrelations. In W. J. Friedman (Ed.), *The developmental psychology of time.* New York: Academic Press.

Wohlwill, J. F. (1966). Piaget's theory of the development of intelligence in the concrete operations period. *American Journal of Mental Deficiency Monograph Supplement, 70,* 57-83.

Wolfe, J. M. (1994). Guided search 2.0: A revised model of visual search. *Psychonomic Bulletin and Review, 1,* 202-238.

Wood, D., & Middleton, D. (1975). A study of assisted problem solving. *British Journal of Psychology, 66,* 181-191.

Wood, D. J., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry, 17,* 89-100.

Wood, D. J., Shadbolt, N. R., Reichgelt, H., Wood, H., & Paskiewicz, T. (1992). EXPLAIN: Experiments in planning and instruction. *AISB Quarterly, 81,* 13-16.

Wood, D. J., Wood, H., & Middleton, D. (1978). An experimental evaluation of four face-to-face teaching strategies. *International Journal of Behavioural Development, 1,* 131-147.

Young, R. M. (1973). *Children's seriation behaviour: A production-system analysis.* Unpublished doctoral dissertation, Carnegie-Mellon University.

Young, R. M. (1976). *Seriation by children: An artificial intelligence analysis of a Piagetian task.* Basel: Birkhauser.

Young, R. M., & O'Shea, T. (1981). Errors in children's subtraction. *Cognitive Science, 5,* 153-177.

Appendix A - Coding scheme for behaviour on the Tower task

The coding schemes used mean that the coded behaviour is a simplification of the actual behaviour on the task. This is because coding a three dimensional world involves a magnitude more effort for which little benefit exists (because when subjects know how the blocks go together, their behaviour mainly consists of revolving the blocks on a horizontal plane; they rarely rotate blocks on a vertical plane). Therefore the coding scheme devised is intended to be optimal for time and efficiency in both coding the behaviour and analysing the codes.

All timings are recorded to the nearest second. This is because millisecond accuracy takes much longer to code (you have to go through the behaviour frame by frame) and is considered unnecessary (because the main interest is in what behaviour is occurring and the order in which it occurs). Although timing is used in some of the analyses described, timing accuracy of one second will be sufficient to carry out the analyses.

In all codes, only the onset time is recorded for the code. The offset time is the onset time of the following code. The construction codes contain a code followed by an argument (the blocks and/or constructions involved in the coded behaviour). Where more than one block/construction is listed as an argument, each is separated by a comma. An example of the fully coded behaviour of one subject is shown in Appendix B.

Correct and Correct-qc-d-in-qc-e

An operation which furthers progress in assembling the pyramid. That is, one which results in a state which is nearer the goal. This definition therefore includes simple stacking, such that building a quarter of a pyramid constitutes four or five correct operations. Stacking and non-stacking operations ("Correct-qc-d-in-qc-e" and "Correct" respectively) are distinguished because some analyses require exclusion of correct stacking operations. The stacking code is an abbreviation of "quarter circle depression in quarter circle elevation". Note that subjects may include two operations in the one action (such as making a hole-pair and stacking it). In these situations, the operations must be distinguished, the observer deciding which one came first, so that the twenty correct, distinct operations to build the pyramid are preserved. There is a need to stick to the four operations it takes to make and stack a layer (for calculating percentages etc). For example, making a hole-pair, then stacking it, then making a peg-pair, then stacking it, then putting them together to make the layer, should be treated as only four operations

(only one of the stacking operations counting). Note that an incorrect construction that gets disassembled does not count as a correct operation (even though its resultant state is nearer to the goal). This is because it will be treated as a disassembly.

Arguments: The blocks constituting the construction (normally two blocks/constructions).

Considers-correct and Considers-correct-qc-d-in-qc-e

This is a correct operation that is only mentally carried out, not physically. This generally occurs when the blocks are in close proximity where they only need pushing together, yet the subject pauses, staring at the blocks. The blocks or constructions considered never touch each other. Stacking and non-stacking operations ("Considers-correct-qc-d-in-qc-e" and "Considers-correct" respectively) are distinguished since some analyses require exclusion of correct stacking operations.

Arguments: The blocks that the subject considered constructing (normally two blocks/constructions).

Incorrect

Where a construction is made that is incorrect. For this to occur, the blocks have to at least touch each other. For example, putting the half peg of one block into the half hole of a different sized block, or aligning two blocks so that their quarter circles make a semi-circle even though they are not connected by a peg/hole. For incorrect operations we also give the task relevant features that were paid attention to in the fitting of the blocks. For example, two blocks may be put together incorrectly via a peg in hole. If also they were flush on their outer edges, but nothing else was task relevant (i.e. no halfpegs align, no halfholes align, a halfhole is not in a halfpeg, their quarter circles do not align), then this gets coded as an incorrect operation with peg in hole and flush. Since blocks/constructions only need to be touching to be recorded as incorrect, then a peg touching a hole, and facing the hole, can be counted as an incorrect fit with the peg in the hole. Note that if the puzzle is completed, then every incorrect construction should have a corresponding disassembly code. This does not apply in two cases: where a block/construction is fitted incorrectly upside down and is then turned the right way up; and where fitting a subsequent blocks/construction to the incorrect one means that part of it is now correct.

Arguments: The blocks constituting the incorrect construction (normally two blocks/constructions).

Considers-incorrect

This is an incorrect operation that is only mentally carried out, not physically. This occurs in much the same way as for considering a correct operation. Considered incorrect operations also have the features listed, as with incorrect operations. The blocks or constructions considered never touch each other.

Arguments: The blocks that the subject considered constructing (normally two blocks/constructions).

Removes-correct

When a block or construction is removed from a *correct* construction, leaving the block/construction and another construction. For example, removing a block from a three-block construction to leave a single block and a construction pair. In order to leave both a block/construction and another construction, the original construction must consist of at least three blocks.

Arguments: The first argument is the block/construction that was removed, the second is the construction that remains, and the third is the original correct construction before a removal took place.

Removes-incorrect

When a block or construction is removed from an *incorrect* construction, leaving the block/construction and another (possibly correct) construction.

Arguments: The first argument is the block/construction that was removed, the second is the construction that remains, and the third is the original incorrect construction before a removal took place.

Disassembles-correct

When a correct construction is taken apart such that all the blocks in the construction are separated (i.e. no blocks touch any others).

Arguments: The correct construction that was disassembled.

Disassembles-incorrect

When a incorrect construction is taken apart such that all the blocks in the construction are separated (i.e. no blocks touch any others).

Arguments: The incorrect construction that was disassembled.

Note we distinguish between a construction being disassembled into its individual constituent blocks (the Disassembles code), and being disassembled leaving a construction and another block or construction (the Removes code). This is because a disassembly which leaves a construction means a previous state may be being revisited. Post-hoc analyses may want to examine this.

Additional codes for incorrect/considers-incorrect/stacking

The codes for Incorrect and Considers-incorrect are split into two, the first detailing the operation performed (i.e. Incorrect or Considers-incorrect), and the second detailing the task appropriate aspects of the fit so that we have information available to attempt to understand the possible goal of the subject when they make incorrect constructions. The codes describing how the operation was performed are detailed below.

- Qc-aligned: The attempted fit meant the quarter circle (qc) of one block/construction became aligned with the quarter circle of the other block/construction.
- P-in-h: The attempted fit meant the peg of one block/construction was placed in the hole of the other block/construction.
- Hp-in-hh: The attempted fit meant the halfpeg of one block/construction was placed in the halfhole of the other block/construction.
- Qc-d-in-qc-e: The attempted fit meant the quarter circle elevation of one block/construction was placed in the quarter circle depression of the other block/construction.
- Flush: The attempted fit meant the outer edges of the newly made construction were flush.
- Dont-know: There are no task appropriate aspects of the fit.

For example, if we fit the two blocks of the largest size having halfholes (Size6Halfhole-Hole, Size6Halfhole-Peg), and fit them such that their quarter circles are aligned (*qc-*

aligned) and the outer edge of the construction is *flush,* but the peg of one is not in the hole of the other, then this results in a fit with two task appropriate features that could have been attended to: firstly, the quarter circles align; secondly, the blocks are flush on their outer edges. Both of these would be listed in the code, which would be

Incorrect/flush/qc-aligned(Size6Halfhole-Hole,Size6Halfhole-Peg).

Both flush and qc-aligned are listed since we have no way of knowing if the goal of the subject was to align the quarter circles or to make the construction flush, we can only ascertain that when fitted, these were both true.

Appendix B - Coding description for subject 1 of the seven year old's

00:00:02	Considers-incorrect/flush(Size6HalfHole-Peg,Size5HalfHole-Peg)
00:00:07	Incorrect/flush(Size6HalfHole-Peg,Size6HalfPeg-Hole)
00:00:08	Disassembles-incorrect(Incorrect[Size6HalfHole-Peg,Size6HalfPeg-Hole])
00:00:11	Considers-incorrect/flush/qc-aligned(Size6HalfHole-Peg,Size6HalfPeg-Peg)
00:00:14	Correct(Size6HalfHole-Peg,Size6HalfHole-Hole)
00:00:15	Considers-incorrect/p-in-h(Size6HolePair,Size6HalfPeg-Peg)
00:00:16	Considers-incorrect/flush(Size6HolePair,Size6HalfPeg-Peg)
00:00:17	Incorrect/flush/qc-aligned(Size6HolePair,Size6HalfPeg-Peg)
00:00:18	Removes-incorrect(Size6HalfPeg-Peg,Size6HolePair, Incorrect[Size6HalfPeg-Peg,Size6HolePair])
00:00:19	Correct(Size6HolePair,Size6HalfPeg-Peg)
00:00:25	Correct(Size6ThreeBlk-HalfPeg-Hole,Size6HalfPeg-Hole)
00:00:30	Correct-qc-d-in-qc-e(Size5HalfHole-Peg,Size6Layer)
00:00:38	Considers-incorrect/flush/qc-aligned(Size5HalfHole-Peg,Size5HalfPeg-Hole)
00:00:40	Removes-correct(Size5HalfHole-Peg,Size6Layer, Stack[Size5HalfHole-Peg,Size6Layer])
00:00:42	Correct-qc-d-in-qc-e(Size5HalfPeg-Peg,Size6Layer)
00:00:43	Removes-correct(Size5HalfPeg-Peg,Size6Layer,Stack[Size5HalfPeg-Peg,Size6Layer])
00:00:45	Considers-incorrect/hp-in-p(Size5HalfPeg-Peg,Size5HalfPeg-Hole)
00:00:46	Correct(Size5HalfPeg-Peg,Size5HalfPeg-Hole)
00:00:49	Correct-qc-d-in-qc-e(Size5PegPair,Size6Layer)
00:00:53	Considers-incorrect/flush/qc-aligned(Size5HalfHole-Peg,Size5PegPair)
00:00:54	Correct(Size5HalfHole-Peg,Size5PegPair)
00:00:58	Considers-incorrect/flush(Size5HalfHole-Hole,Size5ThreeBlk-HalfHole-Hole)
00:01:01	Correct(Size5HalfHole-Hole,Size5ThreeBlk-HalfHole-Hole)
00:01:05	Incorrect/flush/qc-aligned(Size4HalfHole-Hole,Size4HalfPeg-Peg)
00:01:06	Disassembles-incorrect(Incorrect[Size4HalfHole-Hole,Size4HalfPeg-Peg])
00:01:07	Incorrect/p-in-h/qc-aligned(Size4HalfHole-Hole,Size4HalfPeg-Peg)
00:01:08	Disassembles-incorrect(Incorrect[Size4HalfHole-Hole,Size4HalfPeg-Peg])
00:01:10	Incorrect/flush/hp-in-h(Size4HalfHole-Hole,Size4HalfPeg-Hole)
00:01:11	Disassembles-incorrect(Incorrect[Size4HalfHole-Hole,Size4HalfPeg-Hole])
00:01:12	Incorrect/flush/qc-aligned(Size4HalfHole-Hole,Size4HalfPeg-Hole)
00:01:13	Disassembles-incorrect(Incorrect[Size4HalfHole-Hole,Size4HalfPeg-Hole])
00:01:15	Correct(Size4HalfHole-Hole,Size4HalfHole-Peg)
00:01:16	Correct-qc-d-in-qc-e(Size4HolePair,Size5Pyramid)
00:01:19	Correct(Size4HolePair,Size4HalfPeg-Hole)
00:01:22	Correct(Size4ThreeBlk-HalfPeg-Peg,Size4HalfPeg-Peg)
00:01:25	Correct(Size3HalfHole-Hole,Size3HalfHole-Peg)
00:01:28	Correct(Size3HolePair,Size3HalfPeg-Peg)
00:01:34	Considers-incorrect/flush(Size3ThreeBlk-HalfPeg-Hole,Size3HalfPeg-Hole)
00:01:39	Correct(Size3ThreeBlk-HalfPeg-Hole,Size3HalfPeg-Hole)
00:01:42	Correct-qc-d-in-qc-e(Size3Layer,Size4Pyramid)
00:01:47	Correct(Size2HalfHole-Hole,Size2HalfHole-Peg)
00:01:52	Incorrect/p-in-h(Size2HolePair,Size2HalfPeg-Peg)
00:01:53	Removes-incorrect(Size2HalfPeg-Peg,Size2HolePair, Incorrect[Size2HolePair,Size2HalfPeg-Peg])
00:01:55	Correct(Size2HolePair,Size2HalfPeg-Peg)
00:01:58	Considers-correct(Size2HalfPeg-Peg,Size2ThreeBlk-HalfPeg-Peg)
00:02:00	Considers-incorrect/flush(Size2HalfPeg-Peg,Size2ThreeBlk-HalfPeg-Peg)
00:02:03	Correct(Size2HalfPeg-Peg,Size2ThreeBlk-HalfPeg-Peg)
00:02:05	Correct-qc-d-in-qc-e(Size2Layer,Size3Pyramid)
00:02:06	Correct-qc-d-in-qc-e(Size1Top,Size2Pyramid)

Appendix C - Method for the adult Tower task experiment

Subjects

Five subjects were taken from the Psychology department Subject Database at the University of Nottingham. All of these were unfamiliar with the blocks used in the task. Ages of subjects ranged from 21 to 30, 3 were male and 2 female. All subjects were paid for their participation.

Apparatus and materials

The 21 blocks that make up the pyramid puzzle, a camcorder and a microphone (to record behaviour and verbalisations).

Design

The experiment is in two stages. All subjects participated in both stages. For both stages, the main dependent variable was the time taken to complete the task. Further dependent variables were verbalisations, and the correct and incorrect constructions made (and how these were assembled).

Procedure

Subjects were first sat down at a table to help them feel comfortable in the experimental surroundings (e.g. a camcorder in the room). They were then told that the experiment involved thinking aloud whilst performing a task, and that it would involve two parts; the first enabling them to practice giving verbal reports, and the second being the actual experiment.

The practice involved thinking aloud whilst solving a simple twenty piece child's jigsaw puzzle. The experimenter demonstrated verbalisations on this task to begin with, and then subjects were asked to complete it whilst they verbalised their thoughts. Once the experimenter felt the subject was comfortable thinking aloud, the actual experiment took place.

In stage one of the experiment, subjects were given instructions to read which included the goal of the pyramid puzzle (see Stage one instructions section). Once these instructions were understood, subjects were presented with the blocks, in a pre-specified order, and

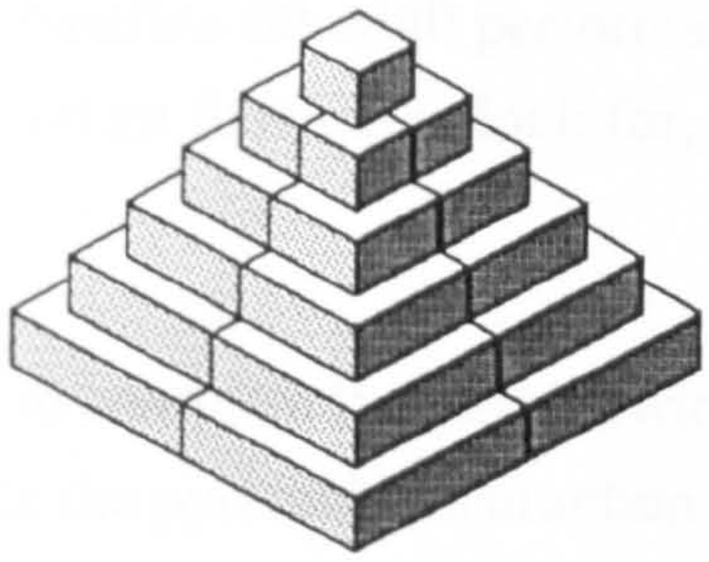
asked to begin. If subjects stopped talking for any length of time (five to ten seconds), the experimenter simply asked them to "keep talking".

Once subjects felt they had completed the task, they were asked to take a seat outside the experimental room whilst the experimenter replaced the blocks into the pre-specified arrangement. Subjects were then given the instructions for stage two (see Stage two instructions section). Once these instructions were understood, subjects were presented with the blocks again, and asked to begin. The experiment finished when subjects felt they had completed the task. Subjects were then thanked for their participation and given an experimental de-briefing sheet (see Experiment de-briefing section).

Stage one instructions

Thank you for participating in this experiment. You are going to be presented with some blocks. Please note that the experimenter is on hand to answer any questions you may have. *If at <u>any point</u> you feel you wish to discontinue the experiment, you may do so.*

The blocks are used to make the construction you see below:



Please use the blocks to produce this construction. Please feel free to take as much time as you want in order to do this. Once you have finished, please inform the experimenter.

Stage two instructions

We now wish you to use the blocks to produce the construction again. You should take as much time as you want in order to do this. Once you have finished, please inform the experimenter.

Experiment de-briefing

We ask you not to discuss this with future subjects, though you may freely discuss this with subjects who have already participated, or non-potential subjects.

The blocks that subjects used can be assembled into a pyramid structure consisting of six layers. Do not worry if you failed to produce this structure.

This blocks puzzle is used to study children's development and look at how different teaching strategies can influence how well children learn. Studies have found that the best teaching method is one which keeps track of the success and failure of the child (e.g. Wood & Middleton, 1975). This "contingent" strategy has the basic rule of "if the child succeeds in performing your instruction then give less help next time, if he fails give more help next time". This is especially so for children of three and four years, although it would seem that for children of five years old and over, any kind of help is equally helpful.

Without teaching, three year old children cannot perform any of the correct operations that are required to build the pyramid. Four year olds produce some, five year olds a bit more, up to eight year olds who can almost complete it by themselves. We wish to examine why this should be the case. Is it a developmental change, is it simply the fact that older children have more experience with blocks, or some other explanation? This experiment will help establish a baseline for adult performance on this task, as well as analysing what are the most important features to look for, and sub-goals to complete, when constructing the pyramid.

The reason subjects were asked to complete the puzzle twice is so that we could examine how quickly they could assemble the pyramid construction having never seen the constituent parts in it, and compare this with the time taken to complete it when already knowing the method for doing so. We also want to ascertain whether different methods were used during the second attempt at the construction, once subjects had already completed the puzzle for their first time.

Thank you again for participating in this experiment.

Appendix D - Method for the seven year olds Tower task experiment

The procedure given here is taken from Wood, Shadbolt, Reichgelt, Wood & Paskiewicz (1992), which contains the precise details of the experiment.

Hardware and software

The EXPLAIN tutor consists of software written in SWI Prolog running on a Sun-4 Sparc workstation. A graphical interface takes in data (child success/failure and materials in use) and displays the next instruction to the helper. The computer is linked to a Sony video disc player and TV monitor. A video-recorder was used to record subjects behaviour.

Task setting

Children were taught individually. On entry, the children were given up to five minutes to play with the 21 randomly scattered blocks before the system was activated. Each child was then given a video-taped instruction followed by instructions based on the contingent tutoring strategy (EXPLAIN gives less help when the child is successful, and gives more help when the child is unsuccessful).

After the Tower had been assembled, it was taken apart again and the child was asked to re-assemble it without help as a post-test.

Appendix E - A production created from proceduralising declarative knowledge

```
(P NEW-DECIDE-FIT-ATTACHMENT1-PRODUCTION159

      =goal>
            ISA                     BUILD-GOAL
            GOAL-TYPE               DECIDE-FIT
            LH-CONSTRN              =BLOCK6PB
            RH-CONSTRN              =BLOCK6PA
            LH-FTR1                 NIL
            RH-FTR1                 NIL
            ORIENT-DIFF             NIL
            GOAL-DONE               NIL
=CONSTRN-FEATURE39>
            ISA                     CONSTRN-FEATURE
            ATTACHED-TO             NIL
            FEATURE                 HALFPEG
            ORIENT-ADJUST           0
            CONSTRN                 =BLOCK6PB
=CONSTRN-FEATURE61>
            ISA                     CONSTRN-FEATURE
            ATTACHED-TO             NIL
            FEATURE                 HALFHOLE
            ORIENT-ADJUST           0
            CONSTRN                 =BLOCK6PA
=ORIENTS9>
            ISA                     FEATURE-FIT-ORIENTS
            FEATURE1                HALFPEG
            FEATURE2                HALFHOLE
            ORIENT-DIFF             =180
      ==>
            =goal>
                  GOAL-DONE         YES
                  LH-FTR1           HALFPEG
                  RH-FTR1           HALFHOLE
```

```
        ORIENT-DIFF             =180
=NEWGOAL159>
        ISA                     BUILD-GOAL
        LH-CONSTRN              =BLOCK6PB
        RH-CONSTRN              =BLOCK6PA
        LH-FTR1                 =CONSTRN-FEATURE39
        RH-FTR1                 =CONSTRN-FEATURE61
        GOAL-TYPE               FIT-TOGETHER
        LH-FIT-REF              NIL
        RH-FIT-REF              NIL
        ORIENT-DIFF             =180
        CALLING-GOAL            =GOAL
        MAIN-GOAL               =GOAL
        !push!                  =NEWGOAL159
)
```