Processing Capacity Defined by Relational Complexity: Implications for Comparative, Developmental, and Cognitive Psychology

Graeme S. Halford
University of Queensland

William H. Wilson
University of New South Wales
and

Steven Phillips
Electrotechnical Laboratory

This work was supported by grants from the Australian Research Council. The authors would like to give special thanks to Keith Holyoak, whose continuing interest in the problems and penetrating observations contributed greatly to the development of this work. We would also like to thank many other colleagues who provided valuable feedback concerning ideas contained in this paper. Special mention is due to John Hummel, Pat Carpenter, Marcel Just, Robert Siegler, John Flavell, Robbie Case, Ken Kotovsky, James McClelland, Alan Baddeley, John Bain, Julie McCredden, Glenda Andrews, Tracey Zielinski, Roger Wales, Doug Saddy, Campbell Dickson, Robert Dennys and Kerry Chalmers, Wayne Wicklegren, Andrew Biemiller, John Tsotsos, Lokendra Shastri, Daniel Berch and one anonymous reviewer.
Author's address: Graeme Halford, Department of Psychology, University of Queensland, 4072, AUSTRALIA, AIRMAIL. Fax: 61 7 3365 4466. Email: gsh@psy.uq.oz.au.
http://www.psy.uq.edu.au/Department/Staff/gsh/

Abstract

It is argued that working memory limitations are best defined in terms of the complexity of relations that can be processed in parallel. Relational complexity is related to processing loads in problem solving, and discriminates between higher animal species, as well as between children of different ages. Complexity is defined by the number of dimensions, or sources of variation, that are related. A unary relation has one argument and one source of variation, because its argument can be instantiated in only one way at a time. A binary relation has two arguments, and two sources of variation, because two argument instantiations are possible at once. Similarly, a ternary relation is three dimensional, a quaternary relation is four dimensional, and so on. Dimensionality is related to number of chunks, because both attributes on dimensions and chunks are independent units of information of arbitrary size. Empirical studies of working memory limitations indicate a soft limit which corresponds to processing one quaternary relation in parallel. More complex concepts are processed by segmentation or conceptual chunking. Segmentation entails breaking tasks into components which do not exceed processing capacity, and which are processed serially. Conceptual chunking entails "collapsing" representations to reduce their dimensionality and consequently their processing load, but at the cost of making some relational information inaccessible. Parallel distributed processing implementations of relational representations show that relations with more arguments entail a higher computational cost, which corresponds to empirical observations of higher processing loads in humans. Empirical evidence is presented that relational complexity discriminates between higher species, is related to processing load in reasoning and in sentence comprehension, and that the complexity of relations processed by children increases with age. Implications are considered for neural net models, and for theories of cognition and cognitive development.

Abstract

It is argued that working memory limitations are best defined in terms of the complexity of relations that can be processed in parallel. Complexity is defined by the number of dimensions, or sources of variation, that are related. Empirical evidence indicates that human adults are limited to processing quaternary relations in parallel. More complex concepts are processed by segmentation (breaking tasks into serially processed components that do not exceed processing capacity) or conceptual chunking ("collapsing" representations to reduce their dimensionality, but at the cost of making some relational information inaccessible). Parallel Distributed Processing implementations show that dimensionality of relations is associated with computational cost, which corresponds to processing load in humans. Empirical evidence is presented that relational complexity discriminates between higher species, and is related to processing load in reasoning for both adults and children. Implications are considered for neural net models, and for theories of cognition and cognitive development.

Keywords: capacity, complexity, working memory, central executive, resource, cognitive development, comparative psychology, neural nets, representation of relations, chunking

Processing Capacity Defined by Relational Complexity: Implications for Comparative, Developmental, and Cognitive Psychology

1.0 We propose that information processing capacity limitations in humans and higher animals should be defined, not in terms of items, but by the complexity of relations that can be processed in parallel. The core argument will be that relational complexity is associated with processing load in human problem solving, with age differences in children's understanding of concepts, and differences between higher animal species. It is not argued of course that relational complexity is the only factor that influences difficulty, because domain expertise, skill with problem solving heuristics, memory availability, and perceptuo-motor factors are also important. Nevertheless, relational complexity is essential to the explanation of some established findings spanning a wide range of literature. We will also explore possible explanations for this limitation in neural net (parallel distributed processing, connectionist) models of cognition. First we will consider previous approaches to working memory, then present our own formulation.

1.1 Processing capacity and working memory

Processing capacity has often been treated as working memory capacity, defined as information that is stored for later processing (Hitch, 1980). Consequently capacity limitations have been defined in terms of number of items, or units of information (Miller, 1956). However storage and processing functions of working memory are partly distinct, because short term storage of information does not necessarily interfere with concurrent processing (Baddeley & Hitch, 1974; Baddeley, 1986; 1990; Halford, 1993; Halford, Bain, & Maybery, 1984; Halford, Maybery, O'Hare, & Grant, 1994; Klapp, Marshburn, & Lester, 1983). Because of this, Baddeley (1986) postulated three systems, a visuo-spatial scratchpad, a phonological loop, and a central executive. The first is concerned with storage of visual-imaginal information, while the second is involved in short-term serial recall or "memory span" tasks. Actual processing is the function of the central executive.

The distinction between information that is stored for later processing, and information that is actually being processed, can be illustrated by the task of mentally adding 79 and 86. The storage of a partial result for later processing that would occur in this task is legitimately called working memory, because the term can include storage for later process. However such storage is distinct from actual processing, in which information constrains a decision, and is not simply stored. For example, when we ask "what number results from adding 9 to 6?" we are not merely storing the addends, but are using them to carry out a computation, and they constrain our decision.

Working memory capacity has also been defined in terms of limits on activation (Anderson, Reder, & Lebiere, 1996; Just, Carpenter, & Hemphill, in press), but this does not provide a general metric for processing complexity. Just et al. (in press) assess complexity in terms of the number of new goals generated, and Case (1985; 1992) uses a similar metric, based on number of embedded goals in a control structure. However, as explained in 6.1.3, the relational complexity metric can subsume the levels of embedding metric and applies more widely. Working memory capacity of children has been assessed using tests that combine processing and storage (Case, 1985; 1992; Pascual-Leone, 1970) but this makes it difficult to assess whether successful prediction is due to processing or storage components of working memory. Anderson et al. (1996) assess processing complexity by the number of symbols in an equation, but this does not necessarily reflect the difficulty of the underlying processes, and does not provide a metric that is general to different types of tasks.

Storage complexity can be measured relatively directly, by number of items or chunks stored as, for example, in memory span tests. Computational complexity of algorithms can also be measured (Garey & Johnson, 1979; Tsotsos, 1990), but as will be discussed in Section 5, it is not clear that this can be translated into a metric for processing complexity in human cognition. The problem can however be approached in the following way. Any cognitive process can be expressed as a function which maps input(s) to output(s). Capacity to perform the process corresponds to capacity to compute the function. A function is a special case of a relation (see 2.3.2), so relational complexity has potential as a measure of processing demand. With processing capacity defined this way, it is not just the number of items or amount of information, but the relations between entities, that is the structure, that is the limiting factor. This was first realised by assessment of the empirical cognition literature, discussed in Sections 3 and 6. However two approaches to modeling higher cognitive processes in neural nets (Halford et al., 1994; Shastri & Ajjanagadde, 1993a) have independently identified a limitation that is consistent with this one.

We are concerned with processing capacity in higher cognitive processes including reasoning, memory operations and language comprehension. Processes such as vision which are performed by modules have higher processing capacity, but are specialised for particular types of input, such as information represented on the retina (Fodor, 1983; Fodor & Pylyshyn, 1988). Modular processes do not impose measurable processing demands (as defined in 2.1), and are not associated with individual differences in intelligence (Anderson, 1992). Modular processes cannot be greatly modified by higher cognitive processes, so thought cannot be used to “re-program” the visual system. By contrast, higher cognitive processes can be modified “on line” and the way a task is performed can be influenced strategically without relearning (Clark & Karmiloff-Smith, 1993). We have attempted to provide a principled account of higher cognitive processes by identifying them with the properties of relational knowledge, defined in 2.2, and we propose that the processing limitations we are considering apply to cognitive functions that have these properties.

2.0 Relations and Processing Demand

The way relational complexity influences processing load can be illustrated by the difficulty of the following sentence:

"The boy the girl the man saw met slept." (1)

The problem here does not reside solely with storage, either of the original sentence or the results of partial processing, but also reflects the amount of information about which decisions must be made. This sentence entails an integrated structure, discussed in 6.1.4, that requires boy, girl, and man to be assigned to roles corresponding to subjects of three verbs and objects of two verbs. Strategies are not generally available to English speakers for serial processing of centre-embedded clauses, and the relative lack of semantic cues or syntactic case markers means people receive little help in deciding which nouns fill which case roles. The result is that we have to decide who saw, who met, and who slept, and identify the objects of "saw" and "met", all together, because we cannot positively identify subjects or objects of any of the verbs until we comprehend the whole sentence. Sentences with this type of reduced relative clause are known to impose high resource demands (Just & Carpenter, 1992; Kimball, 1973).

Another example is provided by Sweller (1993) who analysed the following problem: *Suppose five days after the day before yesterday is Friday. What day of the week is tomorrow?*

Despite our expertise in reasoning about days of the week, this problem is frustratingly difficult. The reason is that, especially in the first sentence, numerous elements are related to each other and cannot be considered meaningfully in isolation. These relations have to be at least partially processed in order to segment the statement into subproblems that can be processed serially. The processing load is felt most keenly when we try to plan this procedure.

The processing load imposed by interacting components of a task can be captured with the concept of relational complexity. We will begin by considering relations between different numbers of factors. At a low level of complexity, consider a case where a cognitive process is constrained by a single factor; for example, our choice of restaurant might depend on the amount of money we have. We can express this as a binary relation, between money and restaurant-choice; that is, a set of ordered pairs, in which each amount of money is associated with a particular restaurant (or with a subset of restaurants). The money-restaurant relation could be modified by another factor, such as importance: The more important the occasion the more expensive our choice of restaurant, though importance might have more influence when we have plenty of money than when we have little. Here we have an interaction between two determining factors. This situation can be represented as a ternary relation, comprising a set of ordered triples in which each amount of money, and each level of importance, is associated with a restaurant. These variables could in turn interact with a third factor, such as hunger, which might make us profligate, but only when we are not really poor. We now have an interaction between three determining factors. This can be expressed as a quaternary relation, comprising a set of ordered 4-tuples, in which each amount of money, level of importance, and state of hunger, is associated with a restaurant choice.

It is clear that the problem becomes more complex as the number of interacting factors increases. This complexity can be measured by the dimensionality of the relation, or number of variables that are related. Problems which entail a binary relation are simpler than those that entail a ternary relation, which are simpler than those which entail a quaternary relation, and so on. The idea of relational complexity is analogous to the number of factors in an experimental design. An experimental design can be thought of as a set of relations between independent and dependent variables. A one-way experimental design is equivalent to a binary relation between one independent and one dependent variable. A two-way experimental design is equivalent to a ternary relation, between two independent and one dependent variables. Experimental designs with more factors permit more complex interactions, but at the cost of more observations (participants) being required. This is analogous to the processing load imposed by problems of high relational complexity.

The complexity of a relation may be defined by the number of arguments. A binary relation has two arguments: For example, the binary relation "bigger-than" has two arguments, a larger and a smaller entity. A ternary relation has three arguments: For example, "love-triangle" is a ternary relation, and has arguments comprising three people, two of whom love a third. Quaternary relations have four arguments, and so on. Each argument can be instantiated in more than one way. For example, each argument of "bigger-than" can be instantiated in an infinity of ways, such as bigger-than(horse,mouse), . . ., bigger-than(whale,dolphin), . . , and so on. Consequently, each argument provides a source of variation, or dimension, and thereby makes a contribution to the complexity of the relation. Our next step is to link relational complexity to the concepts of demand (or load) and resources which have been commonly used to account for performance limitations in cognitive psychology.

2.1. Processing complexity

This may depend on the strategy used by the person in a particular set of circumstances, because different performers use different strategies, and even the same person uses different strategies at different times. The optimum cognitive strategy for real human performers may not correspond to the algorithm that is best theoretically, or to any algorithm that would be used in artificial intelligence, because human cognition operates in ways that are very different in some respects from theoretically optimum algorithms. Strategies can be constrained to some extent by experimental procedures, but if this cannot be done with confidence then the strategy used must be determined by empirical investigation, a point to which we will return in section 6.0. Therefore processing complexity measures need to be specific to the actual cognitive process used.

2.1.1. Complexity of a cognitive process is the number of interacting variables that must be represented in parallel to perform that process. Complexity of processing may also vary over time within one task, so the critical value is the complexity of the most complex step. Tasks can vary in the number of steps they require but this does not necessarily affect processing load, because a task with many steps might impose only a low demand for resources at any one time (e.g., counting peas in a box). Processing demand can be high where steps are embedded in a hierarchical structure, but this entails higher-order relations, and relational complexity is also high (to be discussed in 2.2.5).

2.1.2. Processing complexity of a task is the number of interacting variables that must be represented in parallel to perform the most complex process entailed in the task, using the least demanding strategy available to humans for that task.

2.1.3. Processing demand is the effect exerted by task complexity on a performer, and it reflects the cognitive resources required to perform a task. The core proposal of this paper is that demand is a function of relational complexity. That is, the more interacting variables that have to be processed in parallel, the higher demand will be. Demand is synonymous with "load" and "effort" and the three terms tend to be used interchangeably in the psychological literature. It is the psychological counterpart of computational cost which will be discussed in Section 5. Demand can be manipulated experimentally, with other aspects of the task controlled, and a number of examples of this will be considered in Section 6.

2.1.4. Resources allocated to a task vary as a function of demand and performance. More resources must be allocated to higher demand tasks to maintain performance. The methodology for dealing with demand and resources is now highly developed, and is reviewed by Gopher (1994) and Halford (1993, Chapter 3). Resources utilized can be measured by physiological arousal indicators (Kahneman, 1973), by neural activity assessed using brain imaging techniques (Carpenter & Just, 1996), by subjective feeling of effort assessed through self-report, and by decrement in competing tasks (Navon & Gopher, 1980). Resources invested by an individual in a given task will vary over time as a function of the conditions of performance.

2.1.5. Processing capacity is the limit of resources available. It will vary across individuals and may change over the lifespan (discussed in 6.3). Within a short time frame it is essentially constant, but can be influenced by factors such as physiological state, diurnal rhythms and drugs.

In order to put this argument on a more formal basis, we will consider the nature of relational knowledge.

2.2 Relational knowledge

Given that processing complexity can be related to number of arguments of a relation, the nature of relational knowledge becomes essential to the theory of processing capacity. Phillips, Halford and Wilson (1995; submitted) have argued, on the basis of the work of Codd (1990), that essential properties of higher cognitive processes, including data structures such as lists and trees, can be captured by a model based on processing relations. Our proposed explanation for capacity limitations in higher cognitive processes depends on the complexity of neural net models of relational knowledge, which are considered in Section 4. Therefore we need to specify the properties that relational knowledge must have for a neural net model to be considered adequate. For our present purposes it will be appropriate to say that relational knowledge consists of relational schemas, which we will now define.

2.2.1 Relational schemas are cognitive representations that include elements and relations between elements, and represent situations or activities in the world. In general, an n-ary relation R is a subset of the cartesian product of n sets: $S_1 \times S_2 \times \ldots \times S_n$. Thus if $(a_1,a_2,\ldots,a_n) \in R$ we say that $r(a_1,a_2,\ldots,a_n)$ holds; for example, (cat, mouse) $\in$ LARGER-THAN signifies larger-than(cat, mouse). An n-ary relation comprises a set of n-tuples, where each tuple is a relational instance. We shall refer to R and "larger-than" as relation-symbols. Tuples like $(a_1,a_2,\ldots,a_n)$ and (cat, mouse) we refer to as relational instances. However, it may not be clear, for example, whether (cat, mouse) is being considered as an instance of the relation "larger-than" or of the relation "chases". For this reason, we frequently use the term relational instance for an expression of the form $r(a_1,a_2,\ldots,a_n)$ or larger-than(cat, mouse). Strictly speaking, $r(a_1,a_2,\ldots,a_n)$, larger-than(cat, mouse), and larger-than(mouse, cat) are propositions (see section 2.2.2). Thus we use the term "relational instance" to refer both to the tuple, and to the tuple labelled with the relation symbol. This is not an uncommon practice in cognitive science. Where it would be unclear whether $r(a_1,a_2,\ldots,a_n)$ is being considered as a proposition or a relational instance, we shall refer to (for example) "the relational instance $r(a_1,a_2,\ldots,a_n)$".

2.2.1.1. Representation of a relation requires a symbol to specify the relation R, a representation of the arguments $a_1,a_2,\ldots,a_n$, and a set of bindings between symbol and arguments that maintain the truth of the relation. The binding must constrain the fillers for each argument role so that appropriate members of the cartesian product are bound. For example, in bigger-than(-,-), interpreted in the natural way, the entity in the first argument position must always be bigger than the argument in the second argument position. Thus bigger-than(cat,mouse), bigger-than(mountain,molehill) should be bound but not bigger-than(mouse,cat). The symbol and arguments must retain their identity in the binding. That is, they must not be fused into a whole in which the components cannot be identified.

Representations of relations are valid when they conform to the structural correspondence principle, which holds that relations in the representation must correspond to relations in the world. Formally:

A relational schema comprising elements E^s and relations R^s corresponds to an aspect of the world with elements E^w and relations R^w if there exists a function *f* that assigns each member of E^S in the schema to a member of E^w in the world, in such a way that for any $r^s(e_1^s, e_2^s, \ldots e_n^s)$ in the schema ($r^s \in R^s$, $e_i^s \in E^s$) there is an $r^w(e_1^w, e_2^w, \ldots e_n^w)$ in the world ($r^w \in R^w$, $e_i^w = f(e_i^s)$). These criteria have been specified in more detail by Halford and Wilson (1980) and by Holland, Holyoak, Nisbett and Thagard (1986) and their neural net implementation has been specified by Halford et al. (1994). Structural correspondence is a soft constraint and in some cognitive models it can be overridden by other constraints if of sufficient strength (e.g., Hummel & Holyoak, in press). However learning and induction processes will tend to bring relational schemas into correspondence with the aspect of the world they represent (Holland et al., 1986). This can be done by reducing the strength of representations with inappropriate bindings.

2.2.1.2. Representation of a relational instance $r(a_1,a_2,\ldots,a_n)$ requires a binding between the relation symbol r, and the fillers a_i, each bound to one argument role. Thus bigger-than(whale,dolphin) is a relational instance, whereas the bigger-than relation includes this instance plus appropriate others such as bigger-than(cat,mouse), bigger-than(mountain,molehill) etc.

A relational instance in isolation can be represented by specifying the relation symbol *r* plus each of the role-filler bindings. The relational instance loves(John,Mary) requires a representation of "loves" plus a binding of John to the lover role and Mary to the loved role. We can write this as:

loves + lover.John + loved.Mary

Here the "." symbol signifies the role-filler binding and the "+" serves to concatenate the bindings to the relation-symbol.

This is not sufficient to represent a relation (as distinct from relational instances) because ambiguity occurs as soon as more than one relational instance is represented. Suppose we now add the instance loves(Tom,Wendy), represented as:

loves + lover.Tom + loved.Wendy

When we put the representations of both relational instances together we have:

loves + lover.John + loved.Mary + loves + lover.Tom + loved.Wendy

This represents the fact that John and Tom are lovers and that Mary and Wendy are loved, but it does not distinguish between John loving Mary and John loving Wendy, and is similarly ambiguous with respect to Tom.

The ambiguity can be removed by indicating that John and Mary belong to one instance of loves and Tom and Wendy to another. One way is to index each instance with a unique identifier. This in effect separates (or brackets) the "location" of the representation of each instance. For example, the loves relation could be represented as:

1.(loves + lover.John + loved.Mary) +

2.(loves + lover.Tom + loved.Wendy).

One implication of this method is that, in general, the index does not indicate its contents (the index "1" does not indicate that John and Mary are involved). Thus, potentially all instances must be processed to determine the filler for a given role (in the worst case one may have to search all instances to find the one in which John is the lover of Mary).

An alternative approach is to define each relational instance as a unique n-tuple. This can be done by representing bindings between the relation symbol and the fillers for each argument. For example, we can represent:

loves.John.Mary + loves.Tom.Wendy

This representation is based on <u>symbol-argument-argument</u> bindings, each of which comprises an intact relational instance, together with the symbol for the relation ("loves" in this case). The binding between a symbol and its arguments represents the link specified by the relational instance. Thus representing the tuple loves.John.Mary identifies this relational instance unambiguously, and directly represents the link, identified as "loves", between John and Mary.

2.2.1.3. Relations and working memory. Working memory has often been associated with storage of items of information, as in span tasks, which entail storing strings of items. Therefore it might appear that working memory would store relational instances, that is interconnected sets of items, rather than relations. In some cases this would be sufficient. For example, to understand "John loves Mary" we need only represent relational instance loves(John,Mary). However in some cases working memory entails variables. Consider, for example, reasoning about velocity, defined as $V= s/t$ (where s is distance and t is time). We know that, for example, if we cover the same distance in half the time, velocity is doubled, even though we know no specific values. Thus we are reasoning about the interaction of three variables, velocity, distance and time, and we are dealing with a ternary relation, rather than a relational instance.

2.2.2 A proposition is defined as the smallest unit of knowledge that can have a truth value**.** If the proposition is true, then there will be a corresponding relational instance. For example, the proposition bigger-than(cat, mouse) is an instance of the bigger-than relation. However a proposition need not be true, and a proposition that is false is not a relational instance: bigger-than(mouse,cat) is false, and is not an instance of bigger-than as defined above.

True propositions and false propositions correspond to different subsets of the cartesian product. The relation > is a subset of $S_1 \times S_2$: the subset $\{(a,b) \mid a \in S_1, b \in S_2 \text{ and } a > b\}$. Consider the false proposition >(mouse,cat). The pair (mouse, cat) is not a part of the relation >, (there are no cases of mice being bigger than cats) but the proposition >(mouse,cat) is representable, since (mouse, cat) $\in S_1 \times S_2$. Therefore false propositions can be represented, and correspond to subsets of the cartesian product. Learning and induction will tend to weaken representations of false propositions, and will tend to incorporate semantic constraints. Thus a proposition such as owns(car,Tom) will tend to be excluded. However, false propositions do occur in real cognitive processes and provision must be made for them to be represented.

2.2.3 Truth value of a proposition can be assessed by matching against semantic memory, using a mechanism described in 4.2.1. Truth value can be represented as a higher-order relational instance. For example: false(bark(cats)), meaning that it is false that cats bark. There is a psychological bias to represent true propositions. For example, in mental models theory

(Johnson-Laird, Byrne and Schaeken, 1992) only true contingencies are represented to reduce load on working memory.

Quantifiers are not explicitly represented in relational schemas or in mental models, but the closest psychological property is strength. A strong proposition is one that has a high probability of being true (propositions like "dogs are bigger than cats" that have no definite truth value, in the sense that they are not universally either true or false.[i])

2.2.4 Symbolisation means that a link between the arguments of a relation or relational instance is explicitly symbolised (e.g., one link between "whale" and "dolphin" is explicitly symbolised by "bigger-than"). The link is one of the requirements for identifying a relational instance, which in turn is necessary for a relational instance to be an argument to another relation, thereby forming higher-order relations. It also allows us to distinguish between a number of different links between the same argument sequence.

2.2.5 Higher-order relations and hierarchical structures. Higher-order relations have relational instances as arguments, whereas first order relations have entities as arguments. Higher-order relations can represent connectives; for example implies(>(a,b),<(b,a)) or and(dog(Fido),pet(Fido)). Higher-order relations can be used to represent hierarchical structures, for example: cause(shout-at(John,Tom),hit(Tom,John)).

The repeated variable constraint operates with hierarchical structures. In cause(shout-at(x,y),hit(y,x)) *x* and *y* must be bound to the same entities in both cases. For example: cause(shout-at(John,Tom), hit(Tom,John)) has the required structure but cause(shout–at(John,Tom), hit(John,Tom)) does not. The repeated variable constraint is implemented by ensuring that the first relational instance is in structural correspondence with cause(shout-at(x,y),hit(y,x)). Notice that a mapping that conforms to the structural correspondence principle (in 2.2.1.1) can be formed between them, whereas for the second relational instance such a mapping would be inconsistent (John and Tom must each be mapped to both *x* and *y*).

2.2.6 Omni-directional access means that, given all but one of the components of a relational instance, we can access (i.e. retrieve) the remaining component. For example, given the relational instance mother-of(woman,child), and given mother-of(woman,?) we can access "child", whereas given mother-of(?,child) we can access "woman", and given ?(woman,child) we can access "mother-of".

Omni-directional access is also potentially true of relations. A relational instance $r(a_1,a_2,\ldots,a_n)$ contains *n*+1 objects - the relation symbol and the *n* arguments. Given any *n* components, we can retrieve one or more candidates for the *n*+1st component. However where more than one relational instance is represented, the answers obtained will not always be unique. For example, given the relation "mother-of", the query mother–of(woman,?) may yield child, toddler, infant, baby, teenager, etc. Access may not be equally efficient in each direction. For example, arithmetic addition corresponds to the ternary relation +{ . . (3,2,5), . . ,(5,4,9), . . }. It might be easier to access a sum given two addends, that is +(3,2,?), than to access an addend given the sum and the other addend, that is +(3,?,5), but access in both directions is possible. Having learned our addition tables we can perform subtraction, but perhaps not as efficiently as addition. Another example from the psychological literature is discussed in 6.2.5.1.

<u>2.2.7 Role representation</u> means that relational knowledge entails representation of argument roles or slots, independent of specific instances. Thus bigger-than(-,-) entails representation of roles for a larger and a smaller entity. Roles must be distinguishable but they need not be represented explicitly: the role an argument fills can be identified by its position relative to the other arguments. Thus in "loves.John.Mary" (discussed in 2.2.1.2) we know John is in the lover role by his position in the binding.[ii] The argument positions in a proposition correspond to the sets indicated by their position in the cartesian product. Given the relational instance $r(a_1,a_2,\ldots,a_n)$ for a relational R that is a subset of $S_1\times S_2\times\ldots\times S_n$, each a_i corresponds to a given S_i.

<u>2.2.8 Operations on relations</u> are adapted from those defined in the theory of relational databases (Codd, 1990). They include select, project and join, plus the usual set operators intersection, union and difference (Phillips, et al, 1995). These operations permit information stored in relational knowledge structures to be accessed and manipulated in flexible and powerful ways. The select and project operations together permit access to any element within a relation. Informally, if one thinks of a relation as a table, where rows correspond to relational instances and column names correspond to the role names, then select and project return rows and columns of a table, respectively. The join operation corresponds to combining two tables at a specified pair(s) of columns.

The operators are best described by example. Suppose the relation:

Taller = {(John,Mary),(Mary,Tom)}, where Taller is a subset of Person x Person. Formally, a select operation parameterized with condition C, takes a relation R and returns a new relation R' such that all instances of R' satisfy C. For example, select$_{<Person1,John>}$ Taller -> {(John,Mary)}, returns a single instance binary relation with John at role Person1. A project operation parameterized with attribute (role) name(s) A takes a relation R and returns only arguments at attribute A for each instance in R. For example, project$_{<Person2>}$ Taller -> {(Mary),(Tom)} (i.e., a unary relation with two instances). Taken together, select and project provide omni-directional access to all relational elements.

For example, the query "who is taller than Mary" is realised as:

project$_{<Person1>}$(select$_{<Person2=Mary>}$ Taller)) -> {(John)}, which we write as Taller(-,Mary) -> John, for short.

There are a number of different join operators which take two relations and return a new relation. The outer (or natural) join is analogous to the cartesian product. It returns a relation containing every unique pairwise combination of instances from the argument relations. In this way it permits the construction of higher arity relations - the outer join of two unary relations is a binary relation. Of more interest for our purposes is the equi-join operator, which only joins instances having the same arguments at the specified roles. It provides a way of implementing transitive inference.

For example, equi-join$_{<Person2,Person1>}$(Taller, Taller) -> {(John,Mary,Tom)} is an equi-join of the relation Taller with itself along roles Person2 and Person1. Projecting onto the first and third positions of the resulting relation results in {(John,Tom)} (i.e., the inference "John is taller than Tom"). Finally, since relations are sets the standard set operators intersect, union and difference apply in the usual way.

2.2.9 Decomposability of relations means that relations can be composed of simpler relations. A *decomposable* relation is one which can be written as a conjunct of instances of relations of lower arities. For example, the ternary relation monotonically-larger(a,b,c) can be decomposed into >(a,b) & >(b,c) & >(a,c). This is discussed in more detail in Appendix A. Relations can be decomposed using operators *select* and *project* defined in 2.2.8.

2.2.10 Relational systematicity means that certain relations imply other relations. For example >(a,b) → <(b,a), whereas sells(seller,buyer,object) → buys(buyer,seller,object). Implication can be handled as a higher-order relation as noted in 2.2.5.

2.2.11 An attribute is a relation with one argument. An attribute value is an instance of a unary relation (e.g., ripe(apple23) indicates that apple23 satisfies the unary relation ripe).

2.2.12 Analogy, planning and modifiability. Analogy is a structure-preserving map between a base or source and a target, and representation of relations is at the core of analogies (Gentner, 1983; Gick & Holyoak, 1983; Holyoak & Thagard, 1989). Planning is the main process in strategy development, and entails the organization of a sequence of actions to achieve a goal. Planning has been explicitly modeled by VanLehn and Brown (1980), Greeno, Riley and Gelman (1984) and Halford, Smith, Dickson, Maybery, Kelly, Bain, and Stewart (1995). The development of the strategy is guided by a concept or mental model of the task, which entails representing relations between components of the task.

Higher cognitive processes can be modified “on-line” without necessarily relearning all over again. A performance which distinguishes between higher and lower animal species is the ability to acquire the reversal learning set, that is having learned to choose A in preference to B, the animal must switch to B without relearning (Bitterman, 1960; Bitterman, 1975). If the animal learns the exclusion relation between A and B (i.e. one and only one of A and B is correct) the reversal can be effected by changing the mapping of the stimuli into the relational schema, so the stimulus that was formerly mapped to the positive element of the schema is remapped to the negative element, and *vice verse* (Halford, 1993). Clark and Karmiloff-Smith (1993) have pointed out that modifiability is a criterial attribute of human cognition. Relational representations can achieve this by switching between relations, because when a relation is changed mappings between input and output change. For example, the binary operation of arithmetical addition, a ternary relation, entails a set of mappings between addends and sum, { . . (3,2 → 5), . . (4,7 → 11) . . }. Multiplication entails a different set of mappings { . . (3,2 → 6), . . (4,7 → 28) . . }. A switch to a different relation activates a different set of mappings and modifies the performance.

Relational knowledge is symbolic, content-independent, flexible and modifiable, and can serve the functions of higher cognitive processes. Our next step is to consider the psychological properties that are associated with different levels of relational complexity.

2.3 Psychological interpretations of orders of relational complexity

Each level of relational complexity, from unary to quaternary, corresponds to a distinct category of cognitive tasks. Empirical indicators for each level will be considered in Section 6.

2.3.1 A unary relation has relational instances r(x) that can be interpreted as propositions with one argument, as variable-constant bindings, or as zero-variate functions. A proposition with one argument can represent a state, such as happy(John), an action, such as ran(Tom), an attribute, such as big(dog), or class membership, such as dog(Fido).

2.3.2 Binary relations have relational instances r(x,y) that can sometimes be interpreted as univariate functions; f(a) = b is a special case of a binary relation, in which the mappings are unique; it is a set of ordered pairs, (a,b) such that for each *a* there is precisely one *b* such that (a,b) $\in$ f. A unary operator is a special case of a univariate function; for example, the unary operator change-sign comprises the set of ordered pairs {(x, -x)}.

2.3.3 Ternary relations have relational instances r(x,y,z) that can be bivariate functions, and binary operations. A bivariate function is a special case of a ternary relation. It is a set of ordered triples (a,b,c) such that for each (a,b) there is precisely one *c* such that (a,b,c) $\in$ f. A binary operation is a special case of a bivariate function: a binary operation on a set S is a function from the set S $\times$ S of ordered pairs of elements of S into S; i.e. S $\times$ S $\rightarrow$ S. For example, the binary operation of arithmetic addition consists of the set of ordered pairs of {. . , (3,2,5), . . , (5,3,8), . . , . . }; i.e. {(x,y,z) | x + y = z, x,y,z (say) natural numbers}.

With a ternary relation, it is possible to seek *x* such that r(x,y,z) is a relational instance given *y,z* , and similarly for *y* given *x,z* , or *z* given *x,y*. It thus becomes possible to compute the effects on *x* of variations in *y,z* and so on. This emergence of three-way comparisons in ternary relations is analogous to the emergence of interactions in two-way experimental designs.

2.3.4 Quaternary relations have relational instances of the form r(w,x,y,z). Proportion, a/b = c/d, is a quaternary relation, and entails relations between the four terms, *a,b,c,d*. Given any three terms, plus the knowledge that proportion is entailed, we can predict the remaining term or, given all of *a,b,c,d*, we can decide whether proportion is entailed (a case of omni-directional access, see 2.2.6). With a quaternary relation all the comparisons that are possible with ternary relations can be made, as well as four-way comparisons; the effect on *w* of variations in *x,y,z*, the effects on *x* of variations in *w,y,z,* and so on.

Quaternary relations also encompass trivariate functions and ternary operations. A trivariate function is a special case of a quaternary relation. It is a set of ordered 4-tuples (a,b,c,d) such that for each (a,b,c) there is precisely one d such that (a,b,c,d) $\in$ f. Compositions of binary operations, such as a(b + c) = d correspond to quaternary relations.

2.3.5 Dimensionality of relations: each argument x_i of R (x_1, . . , x_n) for an n-ary relation R can be instantiated in more than one way, and therefore represents a source of variation, or dimension. An n-ary relation may be thought of as a set of points in n-dimensional space, and can represent interaction between *n* variables. The number of arguments, *n*, corresponds to the number of dimensions in the space defined by the relation. This is the basis for our proposed complexity metric. The relation symbol can be predicted in principle from the arguments; for example, given ?(3,2,5), where ? is known to be an arithmetic operation, we know the operation must be addition, whereas given *(?,2,6) we know the first multiplicand must be 3, and so on. Prediction of the relation symbol may depend on constraints, such as knowing the relevant domain, as in this example where it depended on knowing the domain was arithmetic operations. A suitable set of relational instances must be available, and in the worst case all relational instances must be known. Because the relation symbol can be predicted, at least in principle, from the arguments, there are only *n* independent sources of variation in an n-ary relation, and the number of dimensions equals the number of arguments.

Algorithms that embody these properties will be discussed in Section 4. However our next step is to define processing capacity in terms of relational complexity.

3.0 Processing capacity

The amount of information that can be processed in parallel has long been recognized as a critically important datum in cognitive psychology. The most notable attempt to estimate this parameter was made by Miller (1956), who suggested that human capacity was limited to a small number of chunks.

3.1 Chunks

Miller's (1956) concept of a chunk may be defined as a unit of information of arbitrary size, so a digit, an alphabetic character, and an English word may all constitute one chunk, although they vary in information content. The paradox is that the limitation seems to be, not in the amount of information, but in the number of independent units.

3.2 Chunks and dimensions

There is some correspondence between the properties of chunks and the properties of dimensions. Each chunk is a separate signal and fills a separate slot in a message. Attributes or values on different dimensions are at least partly independent of each other, by definition (even nonorthogonal dimensions must convey some independent information, or they are redundant).[iii] Thus chunks, like dimensions, represent units of information that are at least partly independent. Their similarity can be illustrated by the proposition played(John,cricket,oval,Sunday) which has four roles or slots corresponding to player, game, location and day. It seems equally appropriate to regard each filler of these roles as a different chunk, or as a value on a different dimension.

The amount of information (in the information theory sense, Attneave, 1959) conveyed by a chunk depends on the number of alternatives for that slot. For example, "cat" conveys $\mathrm{Log}_2 2 = 1$ bit of information if there are two equally likely alternatives (e.g., cat or dog). If however there were 32 (equally likely) alternatives, the chunk "cat" conveys $\mathrm{Log}_2 32 = 5$ bits. An attribute on a dimension also represents varying amounts of information, depending on the number of alternative values on that dimension. Therefore the amount of information conveyed by a chunk or dimension is arbitrary. Thus both chunks and dimensions are independent units of information of arbitrary size.

3.3. Number of dimensions processed in parallel

Given the link between dimensions and chunks in 3.2, the number of dimensions that can be processed in parallel can be estimated by determining the number of chunks that can be processed in parallel. Miller (1956) proposed that approximately seven chunks were processed in parallel, but difficulties have arisen with his empirical data base (Baddeley, 1986; Halford, Maybery, & Bain, 1988; Schweickert & Boruff, 1986).

More recent research has revised Miller's estimate downwards. Broadbent (1975) examined temporal patterns in recall from semantic memory, and found that items tended to be recalled in groups of approximately three. He suggested that the "magical number seven" proposed by Miller might have reflected the combined output of two systems, each with a capacity of 3-4 items. Fisher (1984) studied visual scanning and found a modal value of four items processed in parallel, with a range of three to five. Halford, Maybery and Bain (1988) assessed the capacity of primary memory, or the information that is currently active (James,

1890) at 4-5 items. A number of other studies reviewed by Schneider and Detweiler (1987) also indicate that approximately four chunks are processed in parallel. Given the identification of chunks with dimensions in 3.2, this implies that approximately four dimensions can be processed in parallel, and humans should be limited to processing quaternary relations in parallel. Most studies indicate a range of values, indicating that this should be a soft limit. Neural net models of relations agree in predicting a soft limit, as will be discussed in Section 5.

An attempt has been made to assess the number of dimensions that can be processed in parallel using interpretation of interactions, because the factors in an interaction cannot be interpreted meaningfully alone, and so there is a constraint to process them in parallel (Halford et al., 1994). An interaction between N factors corresponds to a relation between N independent variables and the dependent variable, as discussed in 2.0, so ability to process four dimensions implies, *prima facie*, understanding of three-way interactions. Academic staff and graduate students who were experienced in interpreting statistical interactions were asked what was the most complex interaction they could interpret unambiguously and with confidence, ignoring scale effects and nonlinearity. Ten answered 2-way, 14 3-way, and 6 4-way. The variations in estimates probably reflect errors due to imprecision of the test, but the mode is 3-way, suggesting four dimensions are processed. While no single study might be definitive there is a degree of consensus across studies with a wide range of methodologies that approximately four dimensions are processed in parallel.

If it should be possible to process two relations in parallel but independently the sum of their processing demands would be less than for a single relation with the same number of arguments. If we consider a k-ary relation R on a set S with s members, then each component x_i in a tuple $(x_1, x_2, \ldots, x_k) \in R$ might be filled in s ways: the number of possible tuples is s^k. Therefore the number of tuples is $2s^2$ for two binary relations but s^4 for a quaternary relation. When we consider neural net representation of relations based on symbol-argument-argument bindings (in 4.1.1.2) the number of binding units is of n^{k+1}, where n is the number of elements of each vector, and a similar argument applies. Therefore the limit will be reached more quickly with a single relation than with two relations having the same total number of arguments. Notice too that more links are defined in a quaternary relation than in two binary relations. Thus R(a,b,c,d) defines links between six pairs ab, ac, . . bc, . . ,cd, whereas R(a,b) and R(c,d) collectively define links between only two pairs, ab and cd.

However it is unlikely that two or more relations can be processed in parallel and independently in the central executive, or within any one system, because they would need to be coordinated to avoid conflict. They could be coordinated by integration into a higher-order relation, but this implies that they are effectively being processed as a single relation. Another way would be to superimpose two or more relations, and this can be done in the neural net models to be discussed in 4.1.1.2. For example the relational instances mother-of(mare,foal), loves(mare,foal), feeds(mare,foal) can all be superimposed. Notice however that superimposing relational instances in this way does not increase the number of dimensions being processed (there are only two arguments in this example). When we consider neural net implementations in 4.1.1 it will be apparent that such superposition adds little computational cost. Superimposed relational instances can be treated as a whole, but they can also be processed separately, using the retrieval process in 4.2.1. On the other hand the relation between them cannot be defined by superposition (e.g., “mother-of”, “loves”, and “feeds” can be fused into a whole equivalent to some kind of composite motherhood concept, or they can be processed as separate relational instances, but no relation is defined between the relational instances).

If four dimensions can be processed in parallel, the next question is how more complex concepts are processed. Many concepts are more complex than quaternary relations, so we must have some means of dealing with these concepts without exceeding our processing capacity.

3.4 Using capacity efficiently

We propose two mechanisms for reducing processing loads imposed by complex concepts. These are conceptual chunking and segmentation.

3.4.1 Conceptual chunking is the recoding of concepts into fewer dimensions. In the limiting, and most typical case, they are recoded into a single dimension. In a mnemonic chunk items function as a unit (e.g., c,a,t becomes a chunk if the three letters form a single word cat). Similarly, elements that are formed into a conceptual chunk function as a whole in a relational structure, and relations between items within the chunk cannot be accessed.

We can illustrate conceptual chunking using the concept of velocity, defined as V= s/t (where s is distance and t is time). The relation between velocity, distance and time is three dimensional, but velocity can be expressed as a single dimension, such as the position of a pointer on a dial; velocity(60 km/h). In this single dimensional representation, no relation is defined between velocity, distance and time. If we want to compute (say) the way velocity varies as distance increases and time decreases, we must return to the three-dimensional representation. Thus conceptual chunks save processing capacity, but the cost is that some relations become temporarily inaccessible. There is also a psychological factor which limits chunking, because experience is required in which there is a constant mapping of elements into chunks (Logan, 1979). Chunking is a form of learning, which takes place over time.

Chunked concepts can be combined with further dimensions to represent higher level concepts, so acceleration can be defined as $A = (V_2 - V_1)t^{-1}$. Acceleration also can be chunked, and then Force (F) can be defined as F = MA (where M = mass). In this way we can bootstrap our way to higher and higher level concepts, without ever exceeding four dimensions processed in parallel. A major function of expertise is to provide ways of chunking that permit the important features of concepts to be represented without imposing excessive processing demands.

Where a role has only one possible filler it can be chunked without loss and the number of dimensions is reduced accordingly. Consider, for example, the relation: mother-of{(Jenny,Tom),(Jenny,Mary),(Jenny,Jill)}. The mother role is always filled by Jenny, so the representation can be collapsed to the unary relation Jenny-mother-of-({(Tom),(Mary),(Jill)}.

The general principles of chunking are: (1) a chunk functions as a single entity, relation-symbol or argument, in a relation, (2) no relations can be represented between items within a chunk, (3) relations between the chunk and other items, or other chunks, can be represented.

In order to assess processing capacity, chunking can be inhibited by using novel structures, for which chunks have not been learned. This does not preclude using familiar domains. For example, in testing transitivity, size relations between unknown persons can be used (e.g., John > Tom, Tom > Peter). Size relations are a familiar domain, but the specific orderings (John, Tom, Peter, etc.) will not have been prelearned as chunks.

3.4.2 Segmentation is breaking tasks into steps which do not exceed processing capacity, and which are processed serially. Examples include algorithms for arithmetic operations, counting, and ordering tasks. Arithmetic algorithms such as multidigit addition generally have to be taught, but there is some degree of autonomy in acquisition of counting and ordering algorithms. It is not possible to determine the precise number of elements in a large set solely by parallel processing, so we use the serial procedure of counting objects one (or, at most, a few) at a time. Children have some understanding of the principles of counting, and this guides the development of their strategies (Greeno et al., 1984). People's concept of an ordered set can provide a mental model for an ordering algorithm (Halford et al., 1995).

The development of counting and ordering strategies illustrate the principle that autonomous development of serial processing strategies requires planning, which depends on representing relations (VanLehn & Brown, 1980). If processing limitations prevent the structure of the task being represented, a strategy cannot be developed without didactic help, which can only be available for a small subset of the cognitive tasks we perform. Thus the "self-programming" property of higher cognitive processes depends on ability to represent relations.

3.4.3 Effective complexity determined by reduction technique

Because complexity can be reduced by conceptual chunking and segmentation, the number of arguments of a relation does not immediately translate into effective complexity. Also, simply increasing the number of arguments by conjunction does not necessarily contribute to the complexity of the resulting relation. The important point regarding relational complexity is the nature of the interaction between the relational elements. Effective relational complexity can be determined using a reduction technique.

More specifically, the effective complexity of a relation is the minimum dimensionality a relation can be reduced to without loss of information. So, if a ternary relation can be reduced to two binary relations without loss of information then effective relational complexity is binary, not ternary. One can determine if a relation can be reduced to a combination of lower order relations by a procedure of decomposing and recombining. If the resulting relation is the same as before then the relation can be decomposed into lower order relations without loss of information. Psychologically, effective relational complexity is the minimum dimensionality to which a relation can be reduced using decomposition and recombination procedures available to human performers.

For example, suppose the following three facts:

1. John played tennis at the school

2. John played soccer at the park

3. Mark played soccer at the park.

Intuitively, it would appear that this domain consists of a ternary relation over Person, Game and Location. That is, Played(Person, Game, Location) = {(John,tennis,school),(John,soccer,park),(Mark,soccer,park)}. So, at first we may claim that the relational complexity of this domain is ternary. However, this ternary relation can be decomposed into two binary relations by splitting the ternary relation along the Game attribute. The resulting relations are: Played(Person, Game) = {(John,tennis),(John,soccer),(Mark,soccer)}; and Is-played-at(Game, Location) = {(tennis,school),(soccer,park)}. Now, recombining these two binary relations by joining them along the common attribute Game results in the original ternary relation Played(Person,Game,Location) containing exactly the same elements. Therefore, the ternary relation is decomposable into two binary relations, and effective relational complexity is binary.

However, suppose now that the domain has changed to include a new fact: 4. Mark played soccer at the school. We will see that inclusion of this fact changes the relational complexity of the domain. The ternary relation Played, consists of elements (John,tennis,school),(John,soccer,park), (Mark,soccer,park) and (Mark,soccer,school). Splitting this relation along the Game attribute results in the two binary relations: Played(Person,Game) = {(John,tennis),(John,soccer),(Mark,soccer)}; and Is-played-at(Game, Location) = {(tennis,school),(soccer,park),(soccer,school)}. However, recombining these two relations results in the new triple: (John,soccer,school), formed by joining pairs (John,soccer) and (soccer, school). However, this triple is not an element of the ternary relation Played(Person, Game, Location) before decomposing, and is not recorded in any of the 4 facts for the new domain. Therefore, decomposing and recombining along the Game attribute has not recovered the original relation, so there has been a loss of information.

The same procedure applied to the Person and Location attributes also results in triples: (John,tennis,park) and (John,soccer,school), respectively, that are not elements of the ternary relation Played(Person,Game,Location). Therefore, this new domain cannot be decomposed into two binary relations, and so its effective relational complexity is ternary. One possible rejoinder to this sort of analysis is to claim that all relations can be decomposed into binary relations by simply creating unique symbol for each element (tuple) of the higher order relation. Under this scheme, the above domain could be decomposed into the single binary relation: Involved(Event,Participant) = {(JTS, John),(JTS, tennis), (JTS, school), (JSP, John), ...}, where JTS is a unique symbol for the event "John played tennis at the school", etc. However, a general encoding scheme requires processing the original ternary relation (e.g., (John, tennis, school) -> JST). Decomposition only becomes effective once the creation process is complete. Less general encoding schemes are possible utilizing only binary relations (e.g., (John, tennis) -> JT), but such schemes are inadequate for relations containing both (John, tennis, school) and (John, tennis, park). Furthermore, its implausible that appropriate encoding strategies are immediately available for novel cognitive tasks.

To relate this example to the analysis of variance analogy, notice that with facts 1-3, location is predicted solely by game (tennis at school, soccer in the park) independently of person. When fact 4 is added person and game jointly predict location, and there are 3 interacting variables.

The reduction checking technique can also be applied to higher-order relations. For example, intuitively one might expect that the relational complexity of transitive inferences (e.g., John is taller than Mary, and Mary is taller than Sue, so John is also taller than Sue) is binary, since such inferences operate over binary relations. However transitive inference is not simply a collection of binary relations. Transitive inferences have the structure: "(A R B) and (B R C), therefore (A R C)", where R is some binary relation and A, B and C are variables ranging over the arguments of R. Transitive inference entails a constraint between two premises and a conclusion, and it is an example of systematicity, as defined in 2.2.10. Reduction analysis shows that the structure of transitive inference is ternary, since it cannot be reduced to a collection of binary relations without loss of information.

The structure of transitive inference can be expressed as a higher-order ternary relation over binary relational instances. That is, Transitive inference(P1,P2,C) = {(aRb, bRc, aRc), (aRb, bRd, aRd), (aRc, cRd, aRd), (bRc, cRd, bRd)}, where P1, P2 and C are the first and second premise, and consequent attribute names (respectively); and *a*, *b*, *c* and *d* are symbols (place holders) to which elements of specific relational instances are aligned. Using the reduction checking technique, we show that transitive inference cannot be decomposed into binary relations.

Suppose we choose to split the relation along the P2 attribute, which results in the two binary relations: And(P1,P2) = {(aRb, bRc), (aRb, bRd), (aRc, cRd), (bRc, cRd)}; and Implies(P2,C) = {(bRc, aRc), (bRd, aRd), (cRd, aRd), (cRd, bRd)}. Rejoining these two relations along attribute P2 results in two ternary relational instances (aRc, cRd, bRd) and (bRc, cRd, aRd), which were not present in the original relation. (That is to say, it is not logically valid to conclude that, for example, Tom is taller than Mark given that John is taller than Bob, and Bob is taller than Mark, if we do not know the relationship between Tom and John or Bob.) Similarly, splitting and rejoining on attributes P1 and C, results in additional relational instances not present in the original relation. Thus, transitive inference is not, in general, decomposable into binary relations.

Indecomposable relations are ultimately significant because, if a relation is indecomposable, then subjects cannot recode the problem by decomposing (ternary) relations into simpler (binary) relations. A case of a indecomposable relation is given in 6.2.4.3. Even where decomposition is theoretically possible, participants might lack the required strategies (algorithms), and the need to cope with more relational instances (of a lower arity) might impose loads of its own (e.g., if higher-order relations are involved, see 6.1.3).

3.5. Effects of processing overload

A participant who cannot construct a representation of the dimensionality required for a task has three options:

1. The concept can be chunked to a lower dimensional representation. This will only be possible if appropriate chunks have been learned or can be constructed, and it results in loss of access to relations between chunked entities.

2. The task can be segmented into smaller steps that are performed serially. This however requires a strategy, autonomous planning of which depends on the participant's ability to represent relations in the task.

3. The participant can default to a lower level representation. This is analogous to performing an experiment with (say) a 3-way design, then analysing the data by a series of 2-way ANOVAs. Just as the analysis would lead to recovery of most of the relevant data in the experiment (all main effects and 2 way interactions would be recovered[iv]), the performance would probably be correct in most respects. However, just as the hypothetical experimenter

would miss the 3-way interactions, our hypothetical performer could not reason about high level relations in the task.

3.6 Capacity and content

It is important to consider whether processing capacity is a function of content. As we indicated in 1.1, the capacity limitations we have defined do not apply to modular processes such as vision. They apply to higher cognitive processes which entail processing explicit relational knowledge, defined in 2.2. However it is still reasonable to ask whether complexity might be influenced by content. We suggest that complexity effects of content variations can be attributed to processes such as conceptual chunking, segmentation, and use of higher order relations. Relations in a familiar domain can be more readily chunked, or higher-order relations may be known which enable the structure to be represented hierarchically, as illustrated in 2.2.5. It can then be segmented by processing one level of the hierarchy at a time, as described in 4.2.5.

An example of content effects is discussed in 6.1.4, but here we will consider two illustrative examples of the way relational complexity can be applied to different task contents and formats. Andrews and Halford (submitted) tested young children's ability to order colored blocks using premises such as "red above green" and "green above blue". In the construction condition the children simply built towers with green above blue, then red above green, and so on. In the prediction condition children had to say in advance which of two blocks, red or blue, would be higher in the tower. The construction condition was easier, apparently because of its concrete, "hands-on" nature. Notice however that in the construction task relations can be processed one at a time: children can first place green on blue, then red on green, etc. By contrast, in prediction they must mentally integrate two relations "red above green" and "green above blue" to yield "red above blue". When number of relations that had to be considered in a single decision was manipulated, with format controlled, it was found that this factor accounted for most of the variance in the task, and there was no significant residual effect of construction-prediction. Thus format was completely subsumed under relational complexity.

The relational complexity metric has been applied successfully to a wide variety of tasks, of which those discussed in Section 6 are a sample. It was applied successfully to children's mathematics by English and Halford (1995). Consider, for example, the following relations between rational numbers: 1/2 = 3/6; 1/2 < 4/6; 1/3 < 3/6; 5/7 > 5/8.

Proportion is a notoriously difficult concept for children learning mathematics, and there seems to be some mystification as to why. However, as this example illustrates, proportions entail a quaternary relation, the variables being the two numerators and two denominators. Therefore the task is likely to be difficult because four dimensions must be processed. This simply illustrates that relational complexity has proved a very serviceable metric for conceptual complexity, in tasks as varied as proportion and ordering blocks.

4.0 Algorithmic Design

The essence of the model is defined at the mathematical (computational) level, and it is designed to account for observed capacity limitations of higher cognitive processes. However research on neural net representation of relations has discovered limitations at least broadly consistent with those observed in psychological data. Integrating the psychological and neural net work on this question has the potential to deepen our understanding of the issue, and to produce more refined questions for future research. This section considers how relations can be represented in neural nets and Section 5 develops the argument that computational cost is a function of relational complexity. Thus the underlying reason for processing capacity limitations may be found in requirements for processing relations in neural nets. However if the reader wishes to avoid the technical complexities of this issue, at least for a first reading, then it is still possible to follow the paper by skipping to Section 6.

The representation of relations in neural nets is currently the subject of extensive research, but even models which differ in architecture are in reasonable agreement about the nature of capacity limitations.

4.1 Neural net models of relational knowledge can be categorised in two ways, type of binding and type of architecture. The models of Hummel & Holyoak (in press), Plate (1995), Shastri and Ajjanagadde (1993a) and Smolensky (1990) use role-filler bindings, while the model of Halford, et al. (1994) uses symbol-argument-argument bindings (defined in 2.2.1.2). Architectures can be divided mainly into models based on a product operation, either tensor product (Halford et al., 1994; Smolensky, 1990) or circular convolution (Plate, 1995) and models based on synchronous oscillation (Hummel & Holyoak, in press; Shastri & Ajjanagadde, 1993a). Other networks exist which learn to represent relations, for example, the recursive autoassociative memory (Pollack, 1988) and BoltzCONS (Touretzsky, 1990). However, the large number of training examples needed to learn appropriate representations makes them unsuitable for models of working memory so they are not considered here.

4.1.1 Tensor and convolution models represent bindings by performing some type of product operation on vectors representing bound entities. Tensor and convolution models can use either role-filler bindings or symbol-argument-argument bindings.

4.1.1.1 Role-filler bindings. In the model of Smolensky (1990) the role-filler binding is represented by the outer product of role and filler vectors, whereas in the model of Plate (1995) it is represented by the circular convolution of the vectors. Thus loves(John,Mary) can be represented in essence by $t_r = v_{role1} \otimes v_{John} + v_{role2} \otimes v_{Mary}$ or by $v_R = v_{role1} * v_{John} + v_{role2} * v_{Mary}$ where $\otimes$ and $*$ represent tensor product and circular convolution respectively, and t_r and v_r represents the relation symbol *r* in the tensor and convolution models respectively. A tensor product net that can represent a role-filler binding is shown in Figure 1A, with an arithmetic example in Figure 1B. In these models all vectors representing roles are superimposed on a single set of units, and vectors representing fillers are superimposed on another set of units. A circular convolution of the vectors in Figure 1B is shown in Figure 1F. A circular convolution is like a compression (technically a projection) of the tensor product matrix, computed by adding along the curved lines as shown. For a lucid explanation of circular convolution see Plate (1995). The elements within the matrix are the *binding units* and their activations are computed in one shot, rather than by incremental adjustment over trials as occurs in learning algorithms. Therefore the matrix represents a dynamic binding in the sense that it represents the currently activated representation, rather than a product of past learning.

Insert Figure 1 here

Using circular convolution the number of elements is constant (as illustrated in Figure 1F); for example, the number of elements in each of v_{role1} and v_{John} is the same as the number in $v_{role1} * v_{John}$, whereas the tensor product of vectors with *n* and *m* elements contains *nm* elements (as illustrated in Figure 1A and 1B). The implications of this for computational cost will be discussed in 5.2.3.

Retrieval from circular convolution representations is noisy (Plate, in press) and requires a cleanup memory, whereas tensor product representations produce an unambiguous output, provided all vectors used form an orthonormal basis (that is, they form a basis for the vector space they span, their lengths are 1, and inner products of distinct vectors are all zero). Though orthonormal vectors are convenient they do not enable crosstalk (interference between similar representations, or similar tasks) to be modeled. This can be done using sparse random vectors, in which similar entities share some units (Wilson, Street, & Halford, 1995). There is then the possibility of confusion and crosstalk.

4.1.1.2 Symbol-argument-argument bindings were illustrated in 2.2.1.2. With this type of model a relational instance is effectively represented by computing the outer product of symbol and argument vectors (Halford et al., 1994). A collection of relational instances can be superimposed on the same representation, by adding up the outer products. Thus representations of loves(John,Mary) can be represented as $V_{loves} \otimes V_{John} \otimes V_{Mary}$ and loves(Tom,Wendy) can be represented as $V_{loves} \otimes V_{Tom} \otimes V_{Wendy}$. These representations can be superimposed by summing the outer products, yielding $V_{loves} \otimes V_{John} \otimes V_{Mary} + V_{loves} \otimes V_{Tom} \otimes V_{Wendy}$.

The resulting sum of outer products is referred to as a tensor. Thus the relational instance $r(a_1,a_2,\ldots,a_n)$ would be represented in a tensor product space $V_R \otimes V_1 \otimes V_2 \otimes \ldots \otimes V_n$, where V_R represents alternative relation symbols including *r*, and v_i (i>0) represents concepts appropriate to the *i*th argument position. A unary relational instance r(a) can be represented in a rank 2 tensor product space $V_R \otimes V_1$. A binary relational instance $r(a_1,a_2)$ can be represented in a rank three tensor product $V_R \otimes V_1 \otimes V_2$. The net in Figure 1A represents a unary relation by this method if one vector represents the relation symbol and the other vector represents the argument. Similarly for the arithmetical example in Figure 1B. Binary relations are illustrated by this method in Figures 1C and 1D. The arithmetic examples of outer products in Figures 1B and 1D show that each element in the matrix (rank 2 outer product) is the product of a component from each of the symbol and argument vectors. Arguments to a relation may also be regarded as role-fillers, and "argument" and "filler" are used interchangeably in this context depending on whether symbol-argument-argument or role-filler models are being considered.

Insert Figure 2 about here

Tensor product implementations of relations from unary to quaternary are shown schematically in Figure 2. In each case there is a vector representing the relation symbol and a vector representing each argument. A ternary relational instance $r(a_1,a_2,a_3)$ can be represented in a rank four tensor product space $V_R \otimes V_1 \otimes V_2 \otimes V_3$. Notice that the example used in Figure 2 represents arithmetic addition and multiplication which, as noted earlier, are ternary relations, superimposed on the same tensor product. If 2 addends (multiplicands) are entered in the argument units (using the retrieval procedure in 4.2.1), and the vector representing addition (multiplication) is entered in the symbol units, the output represents the sum (product). Changing the symbol vector changes the relation that is implemented, and is an example of modifiability as discussed in 2.1.12. Our simulations have shown addition and multiplication can be superimposed in this way on a rank 4 tensor product without interference. A quaternary relational instance $r(a_1,a_2,a_3,a_4)$ is represented in a rank five tensor product $V_R \otimes V_1 \otimes V_2 \otimes V_3 \otimes V_4$, the example in Figure 2 being a composition of two binary operations.

In symbol-argument-argument models roles are determined positionally, a type of coding also used in language. This implies that roles do not need to be explicitly represented and role-filler bindings are unnecessary. The role to which an argument is assigned is defined by its position in the representation. In the tensor product implementation, roles are not represented by vectors, but separate sets of units are used for each argument vector. A procedure is required to ensure structural correspondence, so arguments are represented on the correct set of units. The criteria for valid representation mentioned in 2.2.1.1 are sufficient to ensure this.

To illustrate how roles need not be explicit if arguments are defined relative to each other, consider an instance of the ternary relation arithmetic addition, +(3,5,8). Now suppose we want to superimpose another instance +(2,4,6). This can be done using tensor product representations of symbol-argument-argument bindings as mentioned above. If we were to misalign the representations by superimposing +(2,6,4) on +(3,5,8), that is interchanging the second and third arguments, the error would be detected by the tests for structural correspondence (in 2.2.1.1). Thus a valid relational representation can be established without roles being explicitly represented. The arguments are assigned to the correct role position by ensuring that they are correctly related to each other (in the current example this means that they are in the correct order).

4.1.2 Synchronous oscillation models

In synchronous oscillation models units representing a role oscillate in phase with units representing the filler bound to that role, and out of phase with units representing other roles and fillers. The relational instance loves(John,Mary) would be represented by units representing the agent role of loves oscillating in synchrony with units representing John, while units representing the patient role of loves oscillated in synchrony with units representing Mary (see Figure 3). However units representing the agent role (filler) would oscillate out of synchrony with units representing the patient role (filler). The model of Shastri and Ajjanagadde (1993a) utilises synchronous oscillation for role-filler binding, but much of its power comes from additional nodes and connections. The analogy model of Hummel and Holyoak (in press) uses synchronous oscillation to perform mappings between analogs, but much of its power also comes from other systems, including a distributed semantic memory representation.

Insert Figure 3

Relational instances can be superimposed on the representation, as illustrated in Figure 3. Thus kisses(John,Mary) and marry(John,Mary) can be superimposed on the representation of loves(John,Mary), by having corresponding roles of all relational instances oscillate in synchrony. Notice that no additional phases are required for the superimposed instances, and this is analogous to tensor product representations of symbol-argument-argument bindings discussed in 4.1.1.2, where relational instances can be superimposed on the same set of vector spaces. To foreshadow a point to be made in 5.1, Shastri and Ajjanagadde (1993a) have shown that the major limitation is in the number of distinct phases, rather than the amount of information represented in each phase. This corresponds to the limitation in human processing capacity, which is defined by the number of arguments a relation has, rather than by total information processed.

4.1.3 Comparison of models

Tensor product and synchronous oscillation models appear to be equally capable of representing higher cognitive processes, and the similarity of their properties is at first sight somewhat surprising. However Tesar and Smolensky (1994) have proposed that the architectures are formally reducible to one another, the primary difference being that tensor product models use spatial role vectors whereas synchronous oscillation models use temporal role vectors. Another possible explanation is that their similar properties are due to additional features designed to give them the power to simulate higher cognitive processes. As we will see, the similarity of their properties extends to the processing capacity limitations that are inherent in them.

It is important that, in order to account for working memory, a model must deal with relations. As noted in 2.2.1.2 representation of relations by role-filler bindings requires that each relational instance be stored separately, or be uniquely identified. We will now develop this point further by considering a ternary relation, the binary operation of addition. There are three roles, corresponding to the two addends and the sum, which we will represent as a_1, a_2 and s. Using the role-filler approach we could bind numbers to each role, thus:

a_1.2	a_2.3	s.5
a_1.4	a_2.5	s.9
a_1.3	a_2.4	s.7
a_1.5	a_2.2	s.7 and so on.

If we were to represent all addition facts in this way, then every number would be bound to every role, since any number can serve as first or second addend, or as sum. If all these role-filler bindings are entered into the same representation (e.g., by adding the resulting vectors, as in the models discussed above), and without specific identification of the tuples, then we cannot recover role-filler bindings or relational instances. If we ask "what number is bound to the first addend role?" the answer is "every number", and the same is true for the other two roles. Furthermore, we have only stored role-filler bindings rather than relational instances, so there are no links between addends and sum. Thus even the fact that a_1.2 and a_2.3 are associated with s.5 is not represented, so it is not possible to access a component of the instance, given the remaining component. Thus we cannot ask: if the addends are 2 and 3, what is the sum? Suppose, for example, we were to identify first addend roles that are bound to 2. We cannot then determine which of these cases have 3 bound to the second addend role, because no link has been stored between first and second addends. We cannot retrieve the sum, given the first and second addends, for the same reason.

The solution of identifying each relational instance also has its problems, first because individual identification of every relational instance is implausible when the number of instances is very large. A relational instance such as 2+3=5 is identified by its content (e.g., 2+3=?; ?+3=5) rather than by an index, such as a context vector, that identifies the relational instance.[v] It is implausible that every addition fact we know is individually identified. Second, notice also that, even were instance identification to be used, every role-filler binding in a relational instance would require the same identifier. Thus if we identify this instance as add2,3, we must bind the identifier to all three role-filler bindings, thus:

add2,3: (add2,3).a_1.2 (add2,3).a_2.3 (add2,3).s.5

In other words, the context in which the three role-filler bindings are learnt/memorized must be sufficiently stable as to result in the same identification vector across all three role-filler pairs. Notice also that this representation bears a close resemblance to symbol-argument-argument bindings. Furthermore, the identifier increases the computational cost of the representation, and appears to require an additional rank in the outer product (i.e. rank 3 rather than rank 2, as in Smolensky's (1990) model).

Contrast this with representations based on symbol-argument-argument bindings. Omitting the relation symbol (as with role-filler bindings), we would represent the same facts as:

2.3.5

4.5.9

3.5.8 etc.

The computational cost is high for one relational instance (a ternary relation requires a rank 4 tensor, including the relation symbol vector), but there is no increase in cost for further relational instances because they can be superimposed, the tuples are inherently identified by the bindings, the links between addends and sums are represented, and the other properties of relational knowledge are implemented, as explained in 4.2. Thus while the initial cost of symbol-argument-argument bindings is high, their power is considerable.

Although the synchronous oscillation models of Shastri and Ajjanagadde (1993) and the tensor product symbol-argument-argument binding model of Halford et al. (1994) are very different, they have a common property that is important to capacity limitations. This is that they both map dimensions of the relations (as defined in 2.3.5) to separate dimensions of the representation. In the synchronous oscillation model each argument is assigned to a separate phase in the oscillation (as illustrated in Figure 3). In the symbol-argument-argument binding model each argument is assigned to a separate vector space (illustrated in Figure 1C and 1D).

This means that the dimensions of the relation are mapped directly into phases of oscillation, or into vector spaces. The role-filler models based on tensor products[vi] (Smolensky, 1990) or circular convolution (Plate, in press) do not have this property, because all roles are superimposed, as are all fillers (see 4.1.1.1). As we will see in Section 5, models which map dimensions of the relation to dimensions of the representation imply similar capacity limitations.[vii]

4.2. Modeling relational knowledge

The manner in which these models implement the properties of relational knowledge defined in 2.2 will now be considered. The role-filler model of Smolensky (1990), based on tensor products, handles the storage and retrieval of relational instances, but in its original form it does not appear to incorporate the other features of relational knowledge. The symbol-argument binding model (Halford et al., 1994) was based on Smolensky's tensor product formalism, but with modifications and extensions to handle all features of relational knowledge. The circular convolution model of Plate (in press) handles storage and retrieval of relational instances and gives a good account of similarity, but it does not appear to handle conceptual chunking (see 4.2.4) nor does it provide a general solution to systematicity (see 4.2.9). Role-filler binding models based on synchronous oscillation (Hummel & Holyoak, in press; Shastri & Ajjanagadde, 1993a; 1993b) appear to have been designed to incorporate the properties of relational knowledge in section 2.2. We will emphasise those models that have been designed to implement the properties of relational knowledge.

4.2.1 Retrieval of information

Information stored in a tensor memory can be retrieved by representing a question as a tensor product and computing the inner product (dot product) of the question and memory tensors. The query is a partial relational instance and can be expressed as an outer product with one entity deleted. For example given $r(a_1,a_2,a_3)$ stored as part of the rank four tensor T_{pqrs} in the tensor product $V_R \otimes V_1 \otimes V_2 \otimes V_3$, then say, a_3 can be retrieved by computing the generalised inner product $v_r \otimes v_{a1} \otimes v_{a2} \otimes \_ \bullet \mathbf{T}$, where v_r is the vector representing the relation-symbol *r*, v_{a1} is the vector representing the argument a_1, and v_{a2} is the vector representing the argument a_2. Generalised inner products are described in Appendix B: the _ signifies the component of the tensor which is "retrieved" by computing this generalised inner product. Effectively, the query $r(a_1,a_2,?)$ has been used as input to the tensor memory, and a_3 has been obtained as output. The details of how this generalised inner product may be computed are contained in Appendix B, where it is listed as operation (3).

An analogous procedure is specified for synchronous oscillation models by Shastri and Ajjanagadde (1993a, section 4.3).

4.2.2 Truth value of a proposition can be assessed by matching against memory, in what is essentially a recognition process. The proposition bark(cats) can be represented by a rank 2 tensor product, which can be matched against semantic memory by computing a generalised inner product (dot product) of the tensor with the representations in semantic memory (Humphreys, Bain, & Pike, 1989). The relational instance bark(cats) is treated as a query by representing it as the tensor $v_{bark} \otimes v_{cats}$ as shown above and the dot product of this tensor and tensor representations in memory is computed, as in 4.2.1 (the procedure corresponds to operation (0) in Appendix B). This can be done in parallel for superimposed memories. If the product is non-zero, the proposition is recognized. Thus bark(cats) and bark(dogs) would produce zero and non-zero dot products respectively, so the latter is recognised whereas the former is not. For example:

$v_{bark} \otimes v_{cats} \bullet (v_{bark} \otimes v_{dogs} + .. + v_{sing} \otimes v_{birds}) = 0$ but

$v_{bark} \otimes v_{dogs} \bullet (v_{bark} \otimes v_{dogs} + .. + v_{sing} \otimes v_{birds}) > 0$

The procedure defined by Shastri and Ajjanagadde (1993a, section 4.4) provides a way of assessing truth value of a proposition in synchronous oscillation models.

4.2.3 Relation symbols are represented as separate vectors in the vector space V_R. In synchronous oscillation models the symbol can be represented as a unit firing in a separate phase in the oscillation, or by an additional node connected to role and filler nodes.

4.2.4 Conceptual chunking serves to reduce the rank of a tensor product representation of a relation. It can be implemented by convolution, concatenation (illustrated in Figure 1E), superposition (in which vectors representing arguments are added), and by defining a special function that associates an outer product to a single vector. The outer product representation of r(a,b,c) can be reduced to r(a,b/c), by concatenating or convolving vectors *b* and *c* into a single vector. Features of *b* and *c* can still influence the computation of the relation with *a* because activation can be propagated from units in *b* and *c* to *a* (Figure 1E), but the representation functions as a binary relation, and neither the relation between *b* and *c*, nor the three-way relation between *a*,*b* and *c* is directly accessible.

Unchunking can be achieved by differentiating vectors into other vectors. Algorithms for this have been defined in the STAR analogical reasoning model (Halford et al., 1994; Halford, Wilson, & McDonald, 1995; Halford, Wilson, & Phillips, 1996). In general, lower rank representations can be differentiated yielding more complex relations. For example, in Figure 2E, if the vector representing *b/c* were differentiated into separate vectors representing *b* and *c*, and if all four vectors including the relation-symbol vector (not shown) were then appropriately interconnected, as for a Rank 4 tensor product, a ternary relation could be represented.

A chunked representation is wholistic in that features are represented but are not differentiated into dimensions. Many concepts are wholistic initially and progress to dimensional representation (Smith, 1989). This is like unchunking in that it entails differentiation of a vector into two or more vectors, and to representation of the relation between them.

It is unclear how Plate's (1995) circular convolution model would handle conceptual chunking, at least without significant additions. Chunking involves a compression of a relational instance into an unstructured whole, so the relations between components become inaccessible. However the circular convolution is already a compression (a projection of the tensor product) and it is not clear how a further compression that incorporates the psychological properties of conceptual chunking could be achieved. A further problem is that circular convolution relies on component vectors randomly generated from a guassian or uniform distribution. This has the effect that there is no similarity (as measured by the dot product) between chunked and unchunked representations. Thus, for example, features from *b* and *c* would not, in general, could not influence that computation of the relation with *a* in R(a,b/c).

Hummel and Holyoak (in press) represent the equivalent of a chunk in a synchronous oscillation model by having units that represent part or all of a proposition. For example loves(John,Mary) can be represented as features, as roles and fillers, or as an intact proposition. In the latter case it can be an argument to a proposition such as knows(Sam,loves(John-Mary)).

4.2.5 Higher-order relations and hierarchical structures can be modeled by representing higher-order relational instances with chunked lower-order relational instances as arguments. Consider the relational instance;

cause(shout-at(John,Tom),hit(Tom,John)).

The relational instance shout-at(John,Tom), normally represented as a rank 3 outer product in our model, is chunked into a single vector shout-at_1, as described in 4.2.4, and hit(Tom,John) is chunked similarly as hit_1. The higher-order relation cause(shout-at_1,hit_1) is then represented as a rank 3 outer product.

<u>The repeated variable constraint</u> requires that fillers be bound to the correct roles, as pointed out in 2.2.5. The STAR analogy model (Halford et al., 1996) can achieve this by ensuring hierarchical structures are in correspondence. Consider the relational instances;

cause(shout-at(John,Tom),hit(Tom,John)),

cause(shout-at(Mary,Wendy),hit(Wendy,Mary))

These would be represented as chunked relational instances, as described above. The model maps one level of the hierarchy at a time, then moves to another, usually lower, level and recursively matches corresponding arguments of source and target. Thus the model would first map cause(shout-at_1,hit_1) to cause(shout-at_2,hit_2). It would then unchunk shout-at_1 and shout-at_2 and map shout-at(John,Tom) to shout-at(Mary,Wendy). The model has a bias to maintain the mappings of John to Mary and Tom to Wendy when processing other parts of the structure. It would map hit(Tom,John) to hit(Wendy,Mary), consistent with previous mappings, thereby maintaining structural consistency. It would also compute a goodness-of-mapping score that reflects degree of structural correspondence. The score would be higher for this mapping than for the inconsistent mapping;

cause(shout-at(John,Tom),hit(Tom,John)) to;

cause(shout-at(Mary,Wendy),hit(Mary,Wendy))

The model enforces the repeated variable constraint as a consequence of maintaining structural consistency. Because the person bound to the agent role of “shout-at” is bound to the object role of “hit” in the source, this constraint is maintained in the target because of biases in the algorithm to ensure structural correspondence between base and target.

Shastri and Ajjanagadde (1993a, section 4.5) provides a synchronous activation based mechanism that enforces the repeated variable constraint.

4.2.6 Omni-directional access is implemented by the retrieval process in 4.2.1 because a query can be composed of a relational instance with any component missing. Thus a ternary relation represented as $v_R \otimes v_1 \otimes v_2 \otimes v_3$ can be queried by any of the following means:

$v_R \otimes v_1 \otimes v_2 \otimes \_ \bullet v_R \otimes v_1 \otimes v_2 \otimes v_3 = v_3$

$v_R \otimes v_1 \otimes \_ \otimes v_3 \bullet v_R \otimes v_1 \otimes v_2 \otimes v_3 = v_2$

$v_R \otimes \_ \otimes v_2 \otimes v_3 \bullet v_R \otimes v_1 \otimes v_2 \otimes v_3 = v_1$

$\_ \otimes v_1 \otimes v_2 \otimes v_3 \bullet v_R \otimes v_1 \otimes v_2 \otimes v_3 = v_R$

The procedure for answering wh-queries specified by Shastri and Ajjanagadde (1993a, section 4.7) essentially embodies the omni-directional access property.

4.2.7 Role representation

The role that an argument fills can be indicated by its position relative to other arguments, as discussed in 2.2.7, and its implementation in symbol-argument-argument bindings is described in 4.1.1.2. The synchronous oscillation model of Shastri and Ajjanagadde (1993a) uses role-filler bindings as described in 4.1.2.

4.2.8 Decomposability of relations

The relation represented can be decomposed into the derived relations by replacing the vector in any role with a special vector, namely the sum of all the basis vectors used to represent fillers on that axis of the tensor. Thus a representation of the ternary relation R(x,y,z) can be reduced to representation of $R_3 = (x,y)$ by entering this special vector on the units representing z. Such "collapsing" of a representation to a lower rank has been employed in models of memory (Humphreys et al., 1989) and of analogical reasoning (Halford et al., 1994).

In the tensor product representation of an n-ary relation with instances $r(a_1,a_2,\ldots,a_n)$, the effect of variations in any proper subset of $\{a_1,a_2,\ldots,a_n\}$ on the remaining argument(s) can be computed. For example, suppose that one wishes to use the fixed value b in the final role (the a_n role) and consider the induced (n-1)ary relation $R_{a_n=b} = \{(a_1,a_2,\ldots,a_{n-1}) \mid (a_1,a_2,\ldots,b) \in R\}$. This effect can be achieved by clamping the value in the a_n role of the tensor network to be b. This can, of course, be done with any role, not just the a_n role, and can be iterated so that, eventually, any desired set of roles are fixed in this way. For synchronous oscillation models, the representation of partially instantiated relations in Shastri and Ajjanagadde (1993a, section 3.1) effectively decomposes relations in analogous manner.

4.2.9 Relational Systematicity can be handled by using higher-order relations, as in 4.2.5. For example implies(>(a,b),<(b,a)) can be represented by the tensor product of vectors representing implies and chunked representations of >(a,b) and <(b,a). Systematicity is achieved in the synchronous oscillation model of Shastri and Ajjanagadde (1993a, Section 4.2) by connections that ensure that corresponding arguments oscillate in synchrony (e.g., that the first role-representation in >(a,b) oscillates in synchrony with the second role-representation in <(b,a)).

The circular convolution model of Plate (1995) incorporates systematicity, but there is some doubt as to the generality of the procedure used. In order to enable the model to recognize the structural similarity between “Spot bit Jane, causing Jane to flee from Spot” and “Felix bit Mort, causing Mort to flee from Felix” (by contrast with the superficially similar, but structurally dissimilar “Rover bit Fred, causing Rover to flee from Fred”), Plate used contextualised representations. These entailed adding the property “flee-from” to the representation of Spot, Felix, and Rover, and the property “bite-object” to the representation of Jane, Mort, and Fred. This handles some structurally similar higher-order relational instances, but depends on representing dogs as entities people flee from and people as entities dogs bite. This approach lacks plausibility in relational instances such as “Jane smiled at John, causing John to like Jane” because it is implausible that smiling should be part of the representation of Jane (she may not always smile, even at John), or that liking should be part of the representation of John (he may not always like people). The circular convolution model appears to require additional means of representing structure in order to handle systematicity, and the computational cost of these additions is unknown.

4.2.10 Dimensionality of relations

The dimensionality of a relation was defined in 2.3.5 as the number of arguments. In symbol-argument-argument tensor product representations, a separate vector is used for the relation-symbol and for each argument, so rank is one more than dimensionality. We have used the convention of specifying number of arguments by *n* , and we will use the convention of specifying ranks by *k*. Therefore in this type of model $k = n + 1$. The components of a representation are the relation symbol and the argument representations, so the number of components $= k$. Furthermore if k-1 ranks are known there is at least some potential to predict the *k*th rank (illustrated in 2.3.5) so dimensionality = k-1 = n. Even if relations are represented by formalisms other than tensor products, symbol and arguments must be represented independently of each other, or be individually identified, so that they retain their identity when linked (bound) to other components. Note also that, in the context of neural net models, the dimensionality of a relational concept should not be confused with the number of elements in a vector, which is also sometimes referred to using the term "dimension". As noted in 4.1.3 the models of Halford et al., (1994) and Shastri and Ajjanagadde (1993a) map dimensions of relations directly into dimensions of representations, whereas other models do not.

4.2.11 Analogy, planning and modifiability

Analogy can be successfully modeled using the tensor product representations of relations outlined in 4.1 (Halford et al., 1994; 1995; 1996). A sophisticated model of analogy based on synchronous oscillation has been presented by Hummel and Holyoak (in press).

With symbol-argument-argument models based on tensor products, relations can be modified on line by changing the relation symbol, which selects a new set of relational instances. Relational instances are stored as outer products of symbol and argument vectors, and outer products are summed to form a tensor. We will illustrate with arithmetic addition and multiplication. Addition would be stored as $v_{add} \otimes v_2 \otimes v_3 \otimes v_5 + v_{add} \otimes v_3 \otimes v_6 \otimes v_9 + \ldots$, while multiplication would be stored as $v_{mult} \otimes v_2 \otimes v_3 \otimes v_6 + v_{mult} \otimes v_3 \otimes v_6 \otimes v_{18} + \ldots$ Changing the symbol vector from v_{add} to v_{mult} selects a new set of relational instances, and changes the mappings between addends and sum or product.

4.2.12 Strength can be represented by multiplying the outer product representing the relational instance by a scalar, before adding it to the tensor. Typically the scalar would have a value between 0 and 1 indicating how frequently the relational instance is found to be true: for example bigger-than(dog, cat) would have a scalar a little less than 1 to take account of the small minority of dogs (e.g., chihuahuas) that are smaller than cats.

4.2.13 Operations on relations. Operations on relations can be implemented using tensors. For completeness we provide one tensor implementation for each of the relational operators. The simplest implementation assumes local (with all unit values 0 except for a single unit with value 1) argument vector representations. Relaxing this assumption introduces other properties such as cross-talk, but at the expense of not implementing exact analogues of relational operators. Under the local vector assumption, then, the set operators union, intersection and difference are implemented by pairwise addition, multiplication and subtraction (respectively) of binding units with the same index. The upper limit on activation eliminates multiple occurrences of the same element (consistent with set union), and the lower limit prevents subtraction of nonexisting elements (consistent with difference).

The select operator retrieves relational instances satisfying condition C. Since C is an arbitrary boolean expression, the select operator is a very general and powerful operator. For our purposes, however, we consider a restricted version, where C has the form of a conjunction of filler-role pairs: (a_1,A_1) ^ ... ^ (a_m,A_m) (i.e., select with filler a_1 at role A_1 and filler a_2 at role A_2, etc). The corresponding tensor implementation is to compute the outer product of the fillers a_1 to a_m at the specified tensor axes A_1 to A_m, respectively. Axes with unspecified fillers use a special filler vector $I = (1,...,1)$. Thus, the rank of the tensor (T_C) representing the condition C is the same as the rank of the tensor (T_R) representing relation R. Next, we perform a pairwise multiplication T_C . T_R, resulting in a tensor (T_s) representing the selected relational instances.

The project operator returns the relation between components at the specified roles. The equivalent tensor operation is summation onto the corresponding axes. Formally, given a relation R with attributes (roles) $A_1, ..., A_k$ and a corresponding tensor T with axes labeled $A_1, ..., A_k$, then project_A R, is implemented as $\Sigma_{<A1, ..., Ak>-A}$, where A is the list of projected attributes (or tensor axes) and $<A_1, ..., A_k>$-A is the difference of the two lists (i.e., sum onto the axes not in the list of projected attributes). The rank of the resulting tensor is the same as the arity of the projected relation.

Often one wants to cue a k-ary relational memory with k-1 components to retrieve the target at the kth role. At the relational level, the target is retrieved by successive application of the select and project operators. The select operator retrieves instance(s) containing all *k*-1 components at the specified roles, and the retrieved instance(s) is applied to the project operator which returns the target at the *n*th role. At the tensor level, this combination of select and project is realised by taking the inner product of the *k*-1 components with the tensor, resulting in a vector representing the retrieved target(s). This tensor operation is specified in Section 4.2.1.

The outer join of two relations is simply the outer product of the corresponding tensors. The equi-join, however, is more complicated as it requires only joining those instances that share a common argument at the specified roles. Suppose the relation Taller = {(John,Mary),(John,Bob), (Mary,Tom)}, and its corresponding tensor T = $v_{John} \otimes v_{Mary} + v_{John} \otimes v_{Bob} + v_{Mary} \otimes v_{Tom}$ (the relation symbol is not part of the join operation). One way of implementing the equi-join of the Taller relation with itself is to first, cue the tensor T, with a possible argument at the Person1 role (e.g., v_{Mary} . T). This results in the vector v_{Tom}. Then, cue the tensor again, but at the Person2 role (i.e., T . v_{Mary}). This results in the vector v_{John}. Provided one maintains the cue vector v_{Mary} and the two retrieved vectors v_{John} and v_{Tom}, one can construct the tensor representing the equi-join as $v_{John} \otimes v_{Mary} \otimes v_{Tom}$. Other instances are retrieved in the same manner by cueing with different fillers at the joined roles and adding the result to the tensor representing the equi-join.

To summarise section 4, the properties of relational knowledge can be incorporated in neural net models based on either symbol-argument-argument bindings using tensor products, or on role-filler bindings using synchronous oscillation. Representing these properties effectively depends on a number of additional features, but the basic properties of the representations are important. Despite their differences, both types of model have the property that the dimensions of a relation, the symbol and arguments (fillers) are mapped into dimensions of the representation, either vector spaces or phases of oscillation. This means that components of the relation are represented as intact entities, which retain their identity in the binding, and this is a major factor in the computational cost of representing relations.

5.0 Relational complexity and processing load

So far we have been considering properties of human cognitive functioning, with the aim of accounting for processing capacity limitations observed in psychological data. We have defined cognitive complexity intuitively in terms of the number of interacting variables represented in parallel, and have conceptualised it in terms of the number of arguments in a relation. However we wish to explain processing capacity limitations, and two approaches to neural net modeling of relational knowledge have independently identified possible explanations (Halford et al., 1994; Shastri & Ajjanagadde, 1993a). To explain how processing loads are imposed by relations we need to consider computational complexity which refers to the amount of computation required to perform a task.

Complexity analysis at the computational level is a very general, potentially algorithm-independent method of determining the inherent difficulty of a particular problem. The classic results from computer science have been to identify two very broad classes of problems, called P and NP. In the most general terms, complexity of the former is a polynomial function of some measure of the input, while for the latter it is an exponential (or worse) function. Intuitively, an NP-complete problem is intractable, in the sense that the time required by any known algorithm to solve the problem grows explosively with the size of the problem (the *n*). NP-complete problems can be approached in a number of ways including using an approximate/heuristic algorithm, by avoiding large instances of the problem, or by considering only subclasses of the problem. An algorithm-independent analysis is performed by showing that the problem can be transformed in polynomial time to another problem of known complexity. The power of this method was demonstrated by Tsotsos (1990) with respect to vision, by showing that certain problems in vision transform to NP-complete problems.

However while analysis at this level has been successful with vision, it does not seem to capture processing capacity limitations in cognitive tasks such as reasoning and language. The paradox is that while aspects of vision are intractable by this analysis (Tsotsos, 1990), vision tasks do not appear to impose the kind of demand defined in 2.1.3, and which has been observed in higher cognition processes. The computationally complex tasks of vision appear to be performed without measurable processing demands of the kind discussed in 2.1.3, the standard explanation being that the visual system is a module with high capacity for specialised input, as noted in 1.1. By contrast, many computationally simple tasks in higher cognition impose high processing demands. Ordinary arithmetic, for example, requires relatively little computation, but imposes a high cognitive demand on the human performer, and even such computationally simple tasks as transitive inference problems, in which for example >(a,b) and >(b,c) has to be integrated into monotonically-larger(a,b,c) impose a measurable processing load on adult humans. Tsotsos (1990) shows that visual search is inherently complex because of the combinatorial explosion that occurs as the number of elements to be searched increases. However no such combinatorial explosion occurs in ordinary arithmetic operations or transitive inferences. We must seek the explanation for observed processing demands of such tasks in the architectures employed in higher cognitive processes, for which algorithm complexity is more relevant.

Algorithmic complexity analysis considers how many steps, or how much space is required to compute a given problem for a given algorithm. In the former case, complexity depends on the function which links the number of computational steps required to the size (or length) of the input. A linear time algorithm will complete in $\Theta(n)$ steps (i.e. of the order of *n* steps), where *n* is size of problem (e.g. number of inputs). A polynomial-time algorithm will complete in $\Theta(p(n))$ steps for some polynomial $p(n)$. [viii] An exponential-time algorithm will complete in $\Theta(c^{p(n)})$ steps, so the number of steps grows explosively with size of input. Problem complexity is the least of the complexities of all algorithms to solve the problem.

Both synchronous oscillation and tensor product/convolution models predict limitations on the complexity of relational schemas that can be activated in parallel, though the bases of the limitations are somewhat different. We will examine both types of model in an attempt to find explanations for the limitations observed in the psychological data reviewed in 3.3.

<u>5.1 Synchronous oscillation models</u> are limited by the number of distinct oscillations. This is determined by the ratio of the period of oscillation to the window of synchrony (which is related to the duration of peak, and is approximately the maximum temporal spacing between peaks that are recognized as in phase). This ratio is estimated by Shastri and Ajjanagadde (1993a) to be about five, and by Hummel and Holyoak (in press) to be four to six (to illustrate, notice that in Figure 3, approximately 5 distinct oscillations would be possible). Given the criteria for relational knowledge in 2.2, five distinct entities would permit quaternary relations (a relation symbol and four arguments) to be represented without crosstalk. However Shastri and Ajjanagadde suggest that up to 10 entities could be related with crosstalk. Psychological data discussed in 3.3 appears to correspond to Shastri and Ajjanagadde's prediction of capacity without crosstalk, possibly because performance criteria used in experiments (e.g., low error rates to facilitate analysis of latencies) would tend to preclude crosstalk. However, as noted in 4.1.2, the power of models by Shastri and Ajjanagadde, and by Hummel and Holyoak, depend on additional features, and there does not appear to be any way of calculating the cost of these using computational complexity theory.

<u>5.2 Tensor product models</u> entail a computational cost in space and time. We will consider tensor product representations of relations using symbol-argument-argument bindings as proposed in 4.1.1.2, focusing on the process of accessing the kth component of a relation, given the k-1 other components, as described in 2.2.6 and 4.2.6. Then we will consider

role-filler bindings, first based on tensor products to facilitate comparison, and because existing circular convolution models do not appear to incorporate all properties of relational knowledge (as noted in 4.2). Then we will consider circular convolution models insofar as they can be directly compared with tensor product models. Computational cost can be considered from either a parallel or a sequential processor point of view, but the former is more appropriate to emphasise here.

5.2.1 Complexity for symbol-argument-argument bindings

We will consider time complexity then space complexity for the symbol-argument-argument binding models.

5.2.1.1. Time complexity. In the parallel processing model, one assumes that there is a processor for each unit of the vectors representing symbol and argument(s), a processor for each binding unit, and some *addition units*, to be described below. In order to access the kth component of a relational instance, given arguments $a_1,\ldots,a_{k-1}$, it is necessary to propagate the component values to all the relevant binding units (1 step); each binding unit then multiplies its binding memory contents with the values propagated for the $k–1$ arguments - this requires $k–1$ multiplications. Then it is necessary to add up all these products. How long this takes depends on the rank of the tensor *and* on the length of the vectors - let us suppose all vectors are of length n (so that there are n^k binding units in the tensor network). It will be necessary to add up $n^{k–1}$ products to form each component of the symbol output. This is done by the *addition units* referred to above. The most rapid way to add many items with many processors is to cascade the additions (see Figure 4 for a *binary* cascade adder):

Insert Figure 4 here

This arrangement adds 2^3 items in 3 steps. In general m items can be added together in $ceiling(log_b(m))$ steps, where b is the fan-in of the adders (two in the diagram). If we assume enough processors in the addition unit pool, then the addition step requires $ceiling(log_b(n^{k–1}))$ $= ceiling((k–1)log_b(n))$ steps, for a total of

$$k + ceiling((k–1)log_b(n)) \qquad (2)$$

steps. If b is made large enough, then the second term can be made small, though always at least one.

Neurons may have of the order of 10,000 input connections, but as not all connections may be appropriate to the computation, 10,000 should probably be regarded as the upper limit. A two-level cascade of 10,000-to-1 adders permits addition of 100 million binding units, so we could approximate the second term by the constant 2. Thus at most k+2 steps are required to access a missing component of a relational instance.

Similar computations for a sequential implementation to access the kth component of a relational instance, given arguments $a_1,\ldots,a_{k-1}$ yields the expression $(2k\text{-}1)n^k$ for the number of steps. The important finding however is that with full parallel implementation the tensor product representation is not expensive in terms of time (number of steps), but its spatial complexity is quite large, as shown below.

<u>5.2.1.2. Space complexity.</u> The basic requirement is for the n^k binding units of the rank k tensor and the kn input/output units. In addition to this, in a parallel implementation there would be a need for cascade adders for each side of the tensor network. Assuming that a two-stage cascade is adequate (i.e. that $b^2 \geq n^{k-1}$, so that $b \geq n^{(k-1)/2}$), each cascade would use at most $b + 1$ adders, and there would be n cascades per side (one for each component) and k sides - at most $nk(b + 1)$ adders in all. The n^k binding units dominate the space complexity, providing of course that b is not larger than necessary, that is, not significantly larger than $n^{(k-1)/2}$. Therefore the limiting factor with this representation is the number of binding units, which increases exponentially with dimensionality.

The representation of a single relational instance is quite expensive in terms of space, requiring n^k units to represent that instance, but all combinatorially possible other relational instances can then be represented in the same tensor. (Here n is the length of the vectors, and k is the number of vectors; i.e. one more than the number of arguments). Thus superposition does not incur additional computational cost.

To achieve dynamic binding, the binding units must be interpreted as activations as noted in 4.1.1.1, and activations demand processing resources (Just & Carpenter, 1992). Therefore the rank of relations will be limited by resources available, that is by capacity as defined in 2.1.5.

This is a soft limit, because tensor product representations have the property of graceful degradation (Wilson & Halford, 1994). More recent simulations in our laboratory have extended this finding. For example, a rank 5 tensor of side 16 (i.e. n=16, k=5) with up to 93.75 percent of of the binding units deleted, reliably distinguished stored facts (relational instances) from nonfacts. Such a tensor has the same number of active binding units as an intact rank 4 tensor with side 16. Initial results suggest that the robustness increases significantly as the *n* increases. Thus it appears to be possible to simulate a rank *k*+1 tensor with the number of binding units available to an (intact) rank *k* tensor, over part of the range of *k*, but with processing becoming progressively poorer at successively higher ranks.

This is a soft limit, because tensor product representations have the property of graceful degradation (Wilson & Halford, 1994). More recent simulations in our laboratory have extended this finding to tensor product networks of ranks up to 7. For example, a rank 5 tensor of side 16 (i.e. n=16, k=5) with up to 93.75 percent of of the binding units deleted, reliably distinguished stored facts (relational instances) from nonfacts. Such a tensor has the same number of active binding units as an intact rank 4 tensor with side 16. Our results suggest that the robustness depends on the ratio of number facts stored to number of binding units: the lower the ratio the more robust the network. Provided the value of n, the number of components in each representation vector, is reasonably large (32 was typically adequate in our simulations, for tensors of rank 3 and up, and up to 4000 facts stored) it appears to be possible to simulate a rank k+1 tensor with the number of binding units available to an (intact) rank k tensor, for k = 2 to 6 (at least). Another way of looking at this is to say that the apparently very regimented architecture of a tensor product network is not necessary in order to achieve acceptable memory performance, as 85% or more of the binding units can be removed (i.e. caused to have zero output) with impunity.

5.2.2 Complexity for the role-filler method. Relations can be represented using role-filler bindings, as explained in 4.1.1.1, provided relational instances are identified. We will assume this is done using a separate set of units, because of the implausibility of an identifying code. Smolensky (1990) used tensor products, and Plate (1994, Appendix I) used circular convolution. For the purposes of direct comparison with 5.2.1, we calculate the time and space complexity of accessing relations using the tensor product. Again, we consider the case of accessing the kth argument of a k-ary relation given k-1 arguments.

5.2.2.1. Time complexity.

Given that the complex cue has already been composed, then the role-filler method of two major steps: (1) determining the tensor (representing the relational instance) with the highest similarity (dot product) to the complex cue; and (2) retrieving the target from that tensor.

Assuming *k* roles and *n* fillers, then each relational instance requires *nk* binding units. Further, since each relational instance is represented by a separate set of units, the dot product of the cue with each instance can be performed in parallel. Therefore, the time to compute the highest match is: 1 step to propagate the activations of each cue unit to each tensor unit plus 1 step to multiply cue and tensor elements (pairwise multiplication step of the dot product); and at most 2 steps to sum the activations of each multiplication (summation step of the dot product) - assuming a fan-in of at most 10,000 units as for the symbol-argument method. Theoretically, a winner-take-all network can compute the highest match in 1 step assuming exponential functions and complete interconnection between competing units (Yuille & Geiger, 1995). However, with limited fan-in at most 10,000 relational instances can be compared in parallel. Once the winning tensor is determined, it must be reinstantiated into working memory so that the target component can be retrieved. We will assume that this takes 1 step. In all, at most 6 steps are required, for relations of less than 10,000 instances.

The time required to access the target component is: 1 step to propagate the target's role activations to the binding units, 1 step to multiply the role and tensor activations, plus 1 step to sum activations (since we can assume that the number of roles will be small). The total time for the role-filler is 9 steps.

Although the time complexity is independent of the number of roles, we have not considered the time to compose the cue, which is necessarily $O(k)$ steps since each of the k-1 arguments in the cue must be presented serially over the same group of binding units. The time complexity for composing the cue in the symbol-argument method depends on whether the cue arguments are presented to each separate group of units in parallel or in series. The total time to compose the cue and retrieve the target is *k* steps (parallel cues), or 2*k*-2 steps (serial cues). Either way, the time for both methods is low (i.e., linear in *k*).

5.2.2.2 Space complexity

Each relation is represented as the sum of outer products of role and filler vectors. This requires a rank 2 tensor with *n* possible fillers by *k* possible roles. Therefore the number of units required to represent any one relational instance is *nk*. Under the assumption that each relational instance is stored on a separate group of units, the total number of units needed to store the entire relation is |R|*nk*, where |R| is the number of instances in the relation. In addition, we require *nk* units to compute the complex cue, and |R| winner-take-all units to compute the winning instance. Finally, we need nk units to store the retrieved instance, and n units for the retrieved argument. In total, there are $|R|nk + nk + |R| + nk + n = (|R| + 2)nk + n$ units (i.e., $O(|R|nk)$). Clearly, the space complexity depends on the growth in number of instances as a function of n and k. In the worst case ($|R| = n^k$ - all possible instances), the space complexity is $O(kn^{k+1})$. In the best case ($|R| = 1$ - all one instance relations), the space complexity is $O(nk)$. Average case depends on knowing how many instances are likely to be in any one relation. Under the condition that working memory can store at most C then tasks requiring storage of more instances would force a serial strategy. Under this scenario, the savings in space are traded for an increase in time.

5.2.3. Role-filler models using circular convolution

As noted in 4.2.9 Plate's (in press) circular convolution model does not appear at present to incorporate all properties of relational knowledge, and the computational cost of the additional features required cannot be estimated. Nevertheless, we will examine time and space complexity of circular convolution models for the insights that can be obtained.

The time complexity for circular convolution is the same as for the role-filler tensor method, discussed in 5.2.2.1. Assuming appropriately connected units for implementing circular convolution, each role-filler convolution requires 1 step to propagate activation for each argument, with the remaining processing requiring only a constant number of steps. Thus, the time complexity is still O(k).

Space complexity for circular convolution models is more difficult to determine. Plate (1994) used 840 unique role-filler combinations superimposed over 512 units. Thus, circular convolution permits more role-filler pairs than there are units. However, this method assumes a cleanup memory which does not appear to have been implemented as a neural net. To avoid ambiguity, each relational instance needs to be represented on a different set of units (unless we assume an implausible label attached to each relational instance). The required cleanup memory has essentially the same form as the tensor product implementation of role-filler bindings, the complexity of which is discussed in 5.2.2. In the worst case, where most relational instances must be stored, complexity of the role-filler representation is worse than for symbol-argument-argument representations. Plate's (1994) circular convolution model has achieved interesting results, and the approach has a lot of potential, but its restricted ability to implement the properties of relational knowledge, and the requirements of the cleanup memory, mean that in the context of working memory theory the savings in computational cost may be more apparent than real.

5.2.4 Neural net limits on relational complexity

Although there are still some unresolved issues in neural net representation of relations, there is a strong indication that the limit is in the number of dimensions, or number of entities related, rather than in the total amount of information. Shastri and Ajjanagadde (1993a) showed that synchronous oscillation models can represent quite large amounts of information in parallel, but only a small number of distinct entities can be related (they present many complex cases of reasoning without more than three distinct entities being represented in parallel). Tensor product symbol-argument-argument models also imply a limit in the number of distinct entities, rather than in amount of information. In these models computational cost is polynomial in vector size but exponential in the number of dimensions, so the amount of information that can be represented by a single vector is not significantly limited, but the number of vectors that can be bound in one representation of a relation is limited. Both types of neural net models are consistent with psychological data in implying that the limit is in the number of distinct entities that can be related in parallel.

It is natural to ask why these two models share this limitation. The probable reason is that the synchronous oscillation models, and tensor product symbol-argument-argument models have been designed to comprehensively model higher cognitive processes, and consequently incorporate the properties of relational knowledge defined in 2.2. The computational cost is attributable to dimensions of relations being mapped to dimensions of representations, either vector spaces or phases of oscillation, as noted in 4.1.3 and 4.2.13. Thus the computational costs that we have observed are not inherent in specific architectures, but are inherent in processing relational knowledge. The more adequately a model incorporates the features of relational knowledge the more clearly it entails these costs.

Tensor product role-filler binding models (Smolensky, 1990) incorporate some but not all the properties of relational knowledge, and their computational cost depends on both number of arguments and number of instances stored. Importantly, in general they do not map dimensions of relations to dimensions of representations, and so their computational cost is less clearly related to dimensionality of relations. These models are efficient with few relational instances, but their cost relative to symbol-argument-argument binding models increases when many instances are stored. Circular convolution models (Plate, 1995) appear at first sight to avoid computational costs in space because vector size is constant in the number of entities related. However it is not clear that circular convolution models incorporate all properties of relational knowledge, and the cost of the additional features required is unknown. Furthermore, they produce ambiguous output and depend on cleanup memories that store every relational instance. This incurs a major computational cost that depends on the number of arguments and on the number of instances stored, but in the worst case can exceed the computational cost of symbol-argument-argument bindings.

Synchronous oscillation models suggest about five entities can be processed in parallel, and this would permit one quaternary relation to be represented (symbol and four arguments). The empirical literature reviewed in 3.3 indicated quaternary relations are processed in parallel, so there does not appear to be a major disagreement here. Shastri and Ajjanagadde (1993a) suggest up to 10 entities can be related with crosstalk. Our model agrees to the extent that it implies a soft limit on processing capacity, with performance degrading gracefully as processing load increases.

The ability to process relations more complex than quaternary, though with increased error, may be important in creativity, where early ideas are often imprecise and difficult to communicate. Creative thought also probably requires processing of complex relations, because it entails integrating known relations, and producing new relations which we do not yet know how to chunk or segment. It is possible that processing of relations of high dimensionality is important in creativity, but the increased risk of error would make confirmation and explicitation essential.

6.0. Empirical evidence

Processing load should be a function of relational complexity, which should limit the complexity of tasks that can be performed where chunking or segmentation are inhibited, either by task structure (e.g., indecomposable relations), or by experimental manipulation. Performance predictions depend on analyses of the relations processed, and these analyses in turn need to be confirmed empirically. Therefore testing the theory entails three steps:

1. Develop a process model of the task and empirically verify the model. This can entail an extensive programme of developing and testing models.

2. Analyse relations that must be processed. Where possible, apply the reduction technique outlined in 3.4.3 to determine effective relational complexity. If the task entails steps performed serially, the relevant step is the one in which the largest number of relations is processed (i.e. the peak load). Chunking and segmentation must be controlled (see 3.4.1). When analysing tasks, it is useful to think of the number of interacting variables that are processed in parallel in a given step. Features which remain constant do not contribute to complexity because they can be readily chunked.

3. Test predictions derived from (2) by manipulating relational complexity, with other factors controlled. It is necessary to manipulate the information which needs to be processed in parallel in order to make a decision. This sometimes entails preventing serial processing (an example is given in 6.1.4).

6.1 Complexity and processing load

Our purpose in this section is to show how relational complexity analysis may be applied to tasks which are already well understood, and for which reasonably well validated process models already exist. We adopt a "breadth first" approach, with the aim of showing that relational complexity is applicable to a wide range of phenomena in higher cognition, and therefore offers worthwhile generality.

6.1.1 Transitive inference.

Transitive inference has been shown to be a ternary relation by the reduction technique in 3.4.3. This is consistent with a number of well substantiated models (Sternberg, 1980; Trabasso, 1975) which show that transitive inferences are made by integrating the premise elements into an ordered triple. For example, the premises "Tom is smarter than John, John is smarter than Stan" can be integrated into the ternary relational instance monotonically-smarter(Tom,John,Stan). Maybery, Bain, & Halford (1986) showed that premise integration, which entails a ternary relational instance, should impose a higher processing load than premise coding, which entails binary relational instances, such as smarter-than(Tom, Stan). The middle term can be ignored once integration has occurred, so generating a conclusion entails only a binary relational instance smarter(Tom,Stan). Previous models predicted that processing of negatives (e.g., "John is not as tall as Tom") would increase processing load but these models had not predicted the processing load of premise integration because it occurs in every form of the task, and the models were oriented to accounting for task differences.

Maybery, et al. (1986) tested the prediction that premise integration would impose a high processing load using a sentence verification format for transitive inference, with adult participants. Segmented presentation was used, so the second premise did not appear until the participant indicated that first premise had been encoded. When the participant indicated that the second premise was encoded and the premises had been integrated, a target appeared to which the participant responded by pressing one of two buttons indicating whether the target was consistent with the premises. Probe reaction time, saying "beep" to a tone, was used to assess information processing loads. The control task required verification of separate premises without integration; for example, "Tom is smarter than John, Peter is smarter than Stan". Experimental and control tasks were very closely matched in other respects.

Relational complexity theory predicts there should be significantly longer probe reaction time when the probe occurs while the second premise is being processed than during processing of the first premise or the target, but there should be no such effect with the matched nonintegration control task. Maybery et al. (1986) found a significant probe position by integration/nonintegration interaction of this form, and showed that alternative explanations based on response interference and similar processes could not account for the effects. Processing negatives, previously thought to impose high demands, imposed less load than premise integration.

6.1.2. Verifying relations. The prediction that binary relations impose higher processing loads than unary relations is supported by letter-match data. Posner and Boies (1971) showed that in the letter-match task processing load, as indicated by a probe reaction time secondary task, was greater when the second letter was presented. Coding of one letter is equivalent to a unary relation; e.g., letter(c), representing that the stimulus is the letter "c". When the second letter is presented it also must be encoded, but then a binary relation such as same(c,c), or different(c,k), must be represented. The theory accounts for the higher load observed in the comparison task, because coding requires a unary relation, whereas comparison requires a binary relation. Dimensionality does not preclude other factors, such as memory retrieval, contributing to difficulty, as suggested by the finding that name match is harder than physical match (Posner & Boies, 1971).

6.1.3. The Tower of Hanoi (TOH) puzzle is another task for which there are well validated process models, and it is a good example of a task that entails planning, which depends on relational knowledge. TOH comprises three pegs and a variable number of discs. The discs are placed initially on peg A with the largest on the bottom, the next largest above it, and so on. The goal is to move all discs from peg A to peg C, without moving more than one disc at a time, or placing a larger on a smaller disc. Complexity in TOH depends on the levels of embedding of the goal hierarchy, a metric that has been commonly used to assess complexity (Just et al., in press). The more difficult moves require more levels of embedding of subgoals in the goal hierarchy. However the goal hierarchy metric can be subsumed under the relational complexity metric because, as shown in Table 1, moves with more subgoals entail relations with more dimensions of complexity. The first and every fourth move thereafter are shown because it is only these that require planning (VanLehn, 1991).

Insert Table 1 about here

Consider a 2-disc puzzle. To shift disc 2 from A to C, it is necessary to first shift disc 1 from A to B. The main goal is to shift 2 to C (2C) and the subgoal is to shift 1 to B (1B). The goal hierarchy therefore is 2C 1B, and has two levels (see Table 1). However the task can be expressed as a relation:

Prior(shift(2,C),shift(1,B)).

Shift is a relation, so shifting 2 to C can be expressed as shift(2,C). Similarly for shifting 1 to B. The essence of the goal hierarchy is to perform a set of moves in order to perform another move. This can be expressed as the higher order relation “Prior”, the arguments of which are shift; that is Prior(shift_2(-,-),shift_1(-,-)). As with other relations, complexity is a function of the number of dimensions or roles to be filled, four in this example, so the task is *prima facie* 4 dimensional.

Now consider the more complex 3-disc puzzle: In order to shift 3 to C, it is first necessary to shift 2 to B, in order to do which it is necessary to shift 1 to C (Table 1). There are now 3 levels of goals, and the corresponding relations are also more complex:

Prior(shift(3, C),Prior(shift (2,B),shift(1, C)))

There are now 6 roles so the task is 6 dimensional. By similar argument, the first move on the 4-disc puzzle entails four levels of goals, and can be expressed by the relation:

Prior(shift(4,C),Prior(shift(3,B),(Prior(shift2,C),shift(1,B))))

This is 8 dimensional. Thus number of embedded subgoals corresponds to relational complexity, as Table 1 shows. Conceptual chunking and segmentation can be used to reduce complexity, as with other relational tasks. The first move of the 3-disc puzzle can be simplified by chunking discs 1 and 2:

Prior(shift(3, C),shift (1/2,B/C)).

Shift(1/2,B/C) can be unchunked, yielding:

(Prior(shift(2, B),shift(1,C)).

Thus conceptual chunking and segmentation enable the task to be divided into two 4 dimensional subtasks. This captures the recursive subgoaling strategy that underlies successful performance (VanLehn, 1991), and a conceptual chunk of this kind is called a "pyramid".

Just, et al. (in press) have shown that processing resources are related to the number of new goals that have to be generated for a move. Planning only requires representation of new goals, so the relations that correspond to new goals provide a more realistic estimate of dimensionality of a move. These are shown in Table 1 by underlining the new goals at each step. Where number of goals is reduced, relational complexity is reduced correspondingly.

Our estimate that humans are limited to processing approximately 4 dimensions in parallel implies that humans would normally process no more than one goal and one subgoal in a single move. That is they would process one relation of the form (Prior(shift(2,C),shift(1,A)), or Prior(shift(3, C),shift (1/2,B/C). This is consistent with protocol information (VanLehn, 1991, Appendix, pp. 42-47). A number of predictions based on this analysis have been tested with positive results (Loveday, 1995).

The relational complexity metric subsumes the metric based on levels of embedding of a goal hierarchy, because number of levels of embedding can be mapped directly into relational complexity. However relational complexity also applies to tasks that do not entail subgoals, including tasks where decisions can be made in a single step. It therefore has greater generality. Just as importantly, relational analysis gives insights into the kind of decisions that are made to construct the goal hierarchy. For example it enables us to determine how much information is likely to be processed in one step when a decision is made that in order to move 3 to C, 1 and 2 must be moved to B. Notice that TOH can be performed without processing steps more complex than a quaternary relation. Relational complexity also has the advantage that there is extensive developmental data, to be reviewed in 6.2, indicating ages at which children can typically process each level of complexity. This enables predictions to be made about typical successes of children on specific decisions within the TOH task.

6.1.4 Sentence comprehension

Sentences with reduced relative clauses (i.e. without syntactic markers), with a centre-embedded structure, and without semantic cues, make it difficult for most English speakers to identify cases without parsing the whole sentence. We assume that participants normally segment sentences into constituents which are processed serially as far as possible. In all our modeling, in this and other contexts, we have found it a fruitful assumption that participants tend to minimise processing demand, implying that they never process more information in parallel than necessary. However this type of structure tends to preclude serial processing, thereby preventing the processing load from being reduced by segmentation. This logic has been used by Just and Carpenter (1992), and by Henderson (1994) to test processing load predictions from the theory of Shastri and Ajjanagadde (1993a). An example of such a sentence was mentioned in 2.0:

The boy the girl the man saw met slept. (1)

The subjects and objects of the verbs cannot be identified individually (Kimball, 1973), and such sentences are associated with high processing loads (Just & Carpenter, 1992). Because such sentences tend to inhibit serial processing, they can be used to explore our capacity to process relations in parallel. The meaning of the sentence can be expressed in the following propositions;

slept(boy)

met(girl,boy)

saw(man,girl)

There are five roles to be filled, corresponding to subject and object of the verbs:

slept(Subject1)

met(Subject2,Object2)

saw(Subject3,Object3)

The sentence can be parsed by applying a set of rules (not necessarily the only possible set) shown in Appendix C, which collectively constrain a unique parsing, shown as P10. Given that serial processing is effectively inhibited, parsing the sentence amounts to finding an assignment of noun phrases to roles that fits a set of constraints, that correspond to the rules. There are five roles that must be filled, so the task is five-dimensional, and beyond the capacity of most adults, even exceptional individuals finding it at the limit of their powers.

Andrews and Halford (1994) tested these predictions using centre-embedded and right-branching sentences, with reversible content (e.g "The cow followed the horse") or nonreversible content (e.g., "The boy patted the puppy"). Nonreversible sentences reduce the need for parallel processing (e.g., boy can be assigned directly to the subject role and puppy to the object role) whereas in reversible sentences there are no semantic constraints to assist with identification of subject and object. This again illustrates the point in 3.6 that effects of content can operate through the complexity of relations that have to be processed in parallel.

Examples of centre-embedded sentences of each dimensionality, together with the corresponding propositions, follow (centre-embedded and right-branching structures are not distinguishable with one-dimensional sentences):

<u>One-dimensional</u>:

"The dog ran."	Ran(dog)

<u>Two-dimensional</u> (ignoring initial clause in parentheses, which is used to make the centre-embedded structure meaningful, was the same for all two-dimensional sentences, and can be processed before the remainder of the sentence):

"(The boy saw) the dog that the cat chased."	Chase (cat,dog)

<u>Three-dimensional</u>:

"The emu that the kangaroo passed slept." Sleep(emu)

Passed(kangaroo,emu)

<u>Four-dimensional</u>:

"The baker that the fireman introduced the doctor to died."

Die(baker)

Introduce-to(fireman,doctor,baker)

<u>Five-dimensional:</u>

"The clown that the teacher that the actor liked watched laughed."

Like(actor,teacher)

Watch(teacher,clown)

Laugh(clown)

Participants rated sentences for ease of comprehension, with content controlled. Rated difficulty increased monotonically with dimensionality, but this was modified by an interaction with surface form, so dimensionality had a significantly stronger effect with centre-embedded structure. Dimensionality was also modified by reversibility, but not as strongly as by centre-embedded/right-branching structure. The difficulty ratings strongly reflect the number of bindings that had to be processed in parallel. Participants also indicated if they found a sentence to be incomprehensible. Only four- and five-dimensional sentences were judged to be incomprehensible, and 88 percent of judgments applied to five-dimensional sentences. Of these, 96 percent were applied to centre-embedded sentences. Reversibility did not affect comprehensibility.

These results suggest that when serial processing is inhibited by a centre-embedded structure, people have difficulty assigning words to more than four case roles. The fact that the same limitation does not occur with right-branching sentences supports the dimensionality interpretation in preference to an alternative explanation in terms of the repeated variable constraint, because this would apply equally to right branching-sentences.

6.2 Relational complexity and cognitive development

The theory also predicts that children's performance will be poorer where relational complexity is greater. Furthermore, if children's processing capacity, or the efficiency with which they use their available capacity, develops (Case, 1985; Halford, 1993), they should be able to represent concepts of higher relational complexity with increasing age.

6.2.1 Infancy Content-specific representations appear to be possible in infancy. Baillargeon (1987a; 1987b), has shown that 4-5 month olds can represent attributes and position of vanished objects, at least within the immediate spatio-temporal frame. They dishabituate when a rotating drawbridge moves through the space that should have been occupied by a hidden object, suggesting they can represent its position in the apparatus in front of them. They are also sensitive to attributes such as height and compressibility of the vanished object. Such performances are consistent with representation of the object in the immediate spatio-temporal frame. However there is no evidence that semantically interpretable relations are represented, or that inferences go beyond the perceptible properties of objects. For example, there is no evidence that infants infer that an object must have been removed if a drawbridge moves through the space which it should have occupied.

6.2.2 Unary relations appear to be processed at one year of age, as indicated by category membership (Sugarman, 1982; Younger, 1993) and by disappearance of the A not-B error. This paradoxical phenomenon in infant object constancy research (Wellman, Cross, & Bartsch, 1986), can be interpreted as inability to treat hiding place as a variable, reflecting lack of variable-constant binding (a unary relation, see 2.3.1). That is, when an infant has repeatedly retrieved an object from hiding place A, then continues to search for it at A despite having just seen it hidden at B, the infant is treating hiding place as a constant. To treat hiding place as a variable requires representation of the binding between the variable, location, and the object; i.e. location(object$_1$).

Performance on this task deteriorates as a function of delay between hiding and retrieval, and this effect is greater for younger children (Wellman, et al., 1986). This would follow from a representation with the form of a Rank 2 outer product of vectors representing a variable and a constant, if we make the reasonable assumptions that the representation degrades with delay, but becomes clearer with age, so older children can tolerate more degradation before the representation becomes uninterpretable.

6.2.3. Binary relations such as "larger", "more" etc. appear to be well understood by two years of age, even though there may be some confusion as to which relation is referred to by a particular term (Halford, 1982; 1993). Proportional analogies of the form a:b::c:d are frequently based on binary relations, and there is evidence that young children can perform such analogies in familiar domains (Goswami, 1992). Proportional analogies do not require processing quaternary relations: they require processing two binary relational instances belonging to the same relation. Since the relation is constant to both sides of the analogy, only a single binary relation need be considered. For example, in the analogy "mother is to baby as horse is to what?", (mother, baby) identifies the relation mother-of, and mother-of and mother-of(horse, ?) identifies foal.

This point can be illustrated by comparing a proportion with an analogy. A proportion a/b = c/d is a quaternary relation in that the relation between each of a,b,c,d and the other three entities is defined, whereas this is not true for an analogy. Thus 8:4::27:2 is a valid analogy in that the same relation “>“ holds between 8:4 and 27:2, but it is obvously not a proportion. To illustrate it another way, we can define proportion as the quaternary relation proportional(a,b,c,d). Now knowing four of the elements (i.e. any four out of “proportional/non-proportional”, a, b, c, d) we can determine the fifth, as in 2.3.5 (e.g., given 4,8,3,6 we know it is proportional, but 4,8,3,5 is not proportional). However this is clearly not possible with the analogy 8:4::27:2. Thus a so-called proportional analogy bears only a superficial resemblance to a proportion and they differ markedly in relational complexity.

6.2.4. Ternary relations. A number of concepts based on ternary relations have been associated with persistent difficulties for young children. Transitivity and class inclusion are the best-known examples. Attempts at explanation based on stages of development, or on flawed methodology leading to false negatives, have provided many insights and yielded improved assessments, but still leave important sources of difficulty unexplained (Halford, 1989; 1992; 1993).

6.2.4.1. Transitivity has been a source of difficulty for young children, the reasons for which not been wholly explained (Breslow, 1981; Bryant & Trabasso, 1971; Halford, 1982; 1989; 1992; 1993; Thayer & Collyer, 1978; Trabasso, 1977). We suggest that the unrecognized factor is the processing load imposed by premise integration, which also affects adults (see 6.1.1) but has a greater effect at younger ages (Halford, Maybery, & Bain, 1986).

Piaget's (Piaget, 1950) contention that transitivity is a concrete operational task was challenged by Bryant and Trabasso (1971) who trained children in the premises and found above chance performance in 3-4 year olds. However subsequent work suggested that the children may have been given undue assistance in ordering the premise elements (ie. given premises a<b, b< c etc., both children and adults typically integrate the premises into the ordered set {a,b,c, . , }). The elimination of children who failed to learn the premises might have biased the results, because premises would be difficult to learn if they could not be integrated. When these factors were controlled, children under five no longer succeeded (Halford & Kelly, 1984; Kallio, 1982). Evidence for transitive inference, in children, adults, or other animals, only provides evidence of processing ternary relations if participants are not assisted in ordering the premise elements.

More recent work (Pears & Bryant, 1990) has shown that if 4-year olds are given premises in the form of pairs of colored blocks stacked one above the other (e.g., red above green, green above blue, etc.) they can infer the order of blocks in a tower (e.g., red above blue). However Andrews and Halford (submitted) showed that 4-year olds' performance is marginal at best, and is influenced by relational complexity. This work is consistent with the present theory, and with the empirical work of Halford (1984), in showing that when children under five were required to consider two binary relations in a single decision, they had very little success, but they virtually always succeeded when they could process one relation at a time. Children older than five succeeded on both tasks. We again find evidence that relational complexity affects performance when other factors are controlled, and the effect is greater with younger children.

6.2.4.2. Class inclusion entails undertanding that a and a' are included in b (e.g. apples and nonapples are included in fruit), and therefore b>a (assuming a and a' to be nonempty). Like transitivity, class inclusion was regarded by Piaget (1950) as concrete operational, and unattainable before 7-8 years of age There have been alternative explanations, including misapplication of the rule that a set is counted only once (Klahr & Wallace, 1976; Trabasso et al., 1978; Wilkinson, 1976), and misinterpretation of the question as requiring subclass comparison (Grieve & Garton, 1981; Markman & Seibert, 1976; McGarrigle, Grieve, & Hughes, 1978; Shipley, 1979). These issues have been discussed elsewhere and, although some sources of false negatives have been discovered, class inclusion presents a source of difficulty for children under five that has not been fully explained (Halford, 1992; 1993; Halford & Leitch, 1989; Hodkin, 1987).

Class inclusion and the part-whole hierarchy are essentially ternary relations. A class inclusion hierarchy has three components, a superordinate class, a subclass and a complementary class (e.g. fruit, divided into apples and non-apples). More extended hierarchies are obviously possible, but the concept of inclusion necessarily entails a class and its complement being included in a superordinate class. More extended hierarchies can be handled by conceptual chunking or segmentation. For example, the inclusion of apples, bananas, pears etc. in fruit can be handled by chunking bananas, pears etc. in "nonapples". Where there are more than two levels they can be handled by segmenting the hierarchy into sub-hierarchies and processing two levels at a time. Both the models of Hummel & Holyoak (in press) and Halford et al. (1996, 1997) entail this process. However to chunk apples and nonapples would make the concept of inclusion inaccessible. Part-whole hierarchies, which cause difficulties for young children in arithmetic word problems (Cummins, Kintsch, Reussler, & Weimer, 1988; Halford, 1993; Kintsch & Greeno, 1985) are similar, and comprise a whole divided into two complementary parts.

The difficulty children have with these problems supports the hypothesis that children under 4-5 years have difficulty with ternary relations. Furthermore, as with transitivity, relational complexity has been shown to interact with age, children under 5 succeeding when the task required them to consider only one binary relation, but not when they had to integrate binary relations, while older children succeeded in both cases (Halford & Leitch, 1989). The same finding has been made with matrix classification (Halford, 1980).

6.2.4.3. Concept of Mind. Two reviews of children's concept of mind (Astington, 1993; Halford, 1993) have noted phenomena that, though at first they seem anomalous, can be interpreted in terms of relational complexity. Very young children seem to have difficulty understanding that a person can have two representations of an object. For example, the perceived colour of an object may be modified by a coloured filter (the appearance-reality and perspective-taking tasks), or a person's knowledge of an object's whereabouts might depend on whether they know it has been moved since they last saw it (the false-belief task). Flavell, Green & Flavell (1990) have proposed, based on an extensive assessment of the literature, that young children cannot handle two ways of representing an object (e.g., as both blue and white), but they have not explained why young children should be limited in this way. However some recent analyses (Frye, Zelazo, & Palfai, 1995; Halford, 1993; 1996) converge on the same explanation.

The essence of the problem is that children can represent the relation between a person's knowledge (which we call percept) and the properties of an object or situation. This is a binary relation between percept and object-attribute. However they cannot represent the fact that this is conditional on a third variable. Consider for example, a person who sees an object as white (without filter) and as blue (with filter). This entails representing a ternary relation between viewing-condition, object and percept:

Seen-Object2(<condition>,<object-colour>,<percept>)

Instances of this relation would be:

Seen-object2(no-filter,object-white,percept-white)

Seen-object2(blue-filter,object-white,percept-blue)

This is a ternary relation between the condition[ix], the object and the person's representation of the object. Young children seem unable to do this, and seem to represent the relation between an object and one percept only. That is they represent either:

Seen-Object1(object-white,percept-white), or

Seen-Object1(object-blue,percept-blue).

Both of these are instances of a binary relation, and either could be represented alone by a person who could not represent ternary relations. However to represent the fact that a person can see an object in either of two ways entails conditionalising these relational instances on a third variable, which is equivalent to integrating the binary relations into a ternary relation. Notice that the ternary relation is not decomposable in the sense that it is not reducible without remainder to instances of the binary relation Seen-Object1, because this relation does not represent the fact that alternate ways of viewing the same object are conditional on the viewing condition.

The same limitation would occur with false belief, which entails representing the relation between an object and two different representations of its location, one based on knowledge of where it is, the other based on a false belief about its location. For example, a person sees an object placed in a box, then leaves, and the object is shifted to a basket. Young children have difficulty understanding that the person will believe the object to be in the box though it is really in the basket (Wimmer & Perner, 1983). This can be expressed as the ternary relation:

Find-Object(<known-event>,<actual-location>,<believed-location>), instances of which are:

Find-Object(<saw-moved>,<obj-in-basket>,<believe-obj-in-basket>)

Find-Object(<not-seen-moved>,<obj-in-basket>,<believe-obj-in-box>).

The somewhat paradoxical difficulty that young children have with appearance-reality, perspective-taking and false belief can be interpreted in these terms: They readily understand any of the component binary relations, yet they cannot "put the situation together" and integrate two object-percept relations into a single representation (notice that this is analogous to transitivity which entails integrating two binary-relational premises into a ternary relation, see 6.1.1). Young children's apparently anomalous performance in the concept of mind tasks is perfectly consistent with their performance on other tasks that entail ternary relations. There is already evidence suggesting that processing capacity is a factor in children's concept of mind (Davis & Pratt, 1995; Frye et al., 1995) but the predictions offer scope for more empirical work.

6.2.4.4. Contrary evidence. The most recent, and probably strongest challenge to the proposition that children under five have difficulty with ternary relations comes from Goswami (1995). In Experiment 1 she presented 3- and 4-year old children with two sets of three stacking cups. The experimenter indicated a cup (smallest, middle, or largest) in one set, and the child had to identify the corresponding cup in the other set (smallest, middle, or largest respectively). Performance was high, and appears to provide impressive evidence that 3-4 year olds can process ternary relations. However there was little attempt to analyse the processes by which children made their discriminations.

We can perform at least a first analysis using the reported sizes of the cups, as shown in Table 2. As with Goswami's report, the 12 cup-sizes are shown as the values 1-12, the four sets used being: 1,5,9; 2,6,10 etc. Table 2 indicates three types of cases.

Insert Table 2 here

The first is where the corresponding cup can be selected on the basis of *absolute size* (indicated by a "1" in Table 2). For example, if the Experimenter's set is 1,5,9, and the child's set is 2,6,10, and if the Experimenter indicates cup 5, the correct choice for the child is cup 6, because it is in the corresponding ordinal position (middle). However cup 6 is also closest to cup 5 in absolute size. Therefore a child who paid no attention to binary or ternary relations, and judged on the basis of absolute size (a unary relation), would be likely to choose cup 6, which is the correct answer. This is true in every case where a "1" is entered in Table 2. The critical case is Goswami's different-size cup group in the different spatial position condition, because it is only here that cups were neither identical, nor in the same spatial position, in the two sets. This case corresponds to the off-diagonal entries in Table 2. There are 24 out of 36 cases, or 66.67 percent, where the correct cup can be determined on the basis of absolute size. There are a further 8 cases, or 22.22 percent, where absolute size gives two equally likely answers, one of which is correct (indicated by ".5" in Table 2). These would be expected to yield a further 11.11 percent correct answers. Thus the expected performance if children attended only to absolute size, without processing any relations of higher rank than unary, is 78 percent correct. Furthermore some additional correct answers could be obtained by processing binary relations (to be discussed below). Goswami reports 86 percent success for 4-year olds, and 70 percent success for 3-year olds (averaging over the analogy/no-analogy treatment, which is not relevant to our present argument). Thus these superficially impressive performances provide no evidence for processing ternary relations by 3-year olds, and doubtful evidence by 4-year olds.

In Experiment 2 Goswami had 3- and 4-year old children map fractions from one set to another. For example, given three glasses of lemonade, 1/4, 1/2 and full, the child would be asked to see the correspondence between 1/2 glass of lemonade and 1/2 box of chocolates. In the different spatial position condition, the performance of 4-year olds was 87 percent correct, and that of 3-year olds 54 percent. Absolute size matching is not possible in this experiment, but it is necessary to ask what success can be achieved using binary relations. Two elements in an ordered set of three can be chunked, reducing the task to the binary relation larger/smaller; for example 1/4 might be labeled "smaller", whereas 1/2 and full are chunked as "larger" (this is an example of the kind of chunking in Figure 1E). If the correct item is 1/4, ($p = .33$) this leads to 100 percent correct. If the correct item is either 1/2 or full ($p = .67$), it leads to 50 percent. The overall expected percentage correct is therefore $.33 + .67x.50 = .67$. The three-year olds are clearly not above the binary-relation baseline, and no test was made to see whether 4-year olds were significantly above this baseline.

Experiment 3 entailed mapping between levels of loudness, pitch, hardness, height, etc., which did not permit use of absolute cues. There is the possibility of chunking to binary relations as discussed above, but we will not pursue that issue here. The 4-year olds were again successful but, in both Experiments 2 and 3, <u>the mean age of the 4-year olds was 4 years 11 months, range 4.8-5.1</u>. No data for 3-year olds is reported on this task, so the only evidence for mapping ternary relations is obtained from children who are at, or very close to, 5 years old. The data actually <u>support</u> the proposition that ternary relations are first processed at a median age of 5 years.

<u>6.2.4.5 Summary of ternary relations evidence</u>. The fundamental problem here is that cognitive development research must take account of actual cognitive processes to be theoretically meaningful (Halford, 1982; 1989; 1993; Siegler, 1981). The more important evidence however comes from those studies in which relational complexity has been varied while holding other factors constant. It appears that tasks which entail ternary relations are consistently found to cause difficulty for young children, even though they readily process binary relations. This suggests that relational complexity is an important factor in children's cognitive performance, and it offers a solution to the mystery as to why these tasks have seemed unaccountably difficult.

<u>6.2.5 Quaternary relations</u>. Proportion (see 2.3.4) and the balance scale entail quaternary relations, because they entail relations between four variables, and both have been found difficult for young children, but there is more extensive research on the balance scale.

<u>6.2.5.1 The balance scale</u> entails the quaternary relation balance-state(W_l , D_l ,W_r , D_r). The task is difficult for children below about 11 years, and they tend to use lower rank rules, but even adults rarely use the cross products rule without specific instruction (Siegler, 1981; Surber

& Gzesh, 1984). The task can be segmented by, for example, computing the moments on each side then comparing them to see which side will go down, or whether the beam will balance: $W_l \times D_l = M_l$; $W_r \times D_r = M_r$. Each of these steps requires processing a ternary relation. The comparisons entail the rules: $M_l = M_r \rightarrow$ balance; $M_l > M_r \rightarrow$ left side down; $M_l < M_r \rightarrow$ right side down. Each is a comparison of two values and is a binary relation. Therefore the task can be performed by processing ternary relations one at a time. However planning this strategy means being able to represent the fact that the moments are determined by weight and distance on each side (see 2.2.12). This entails representing the ternary relation relation balance-state(W_l , D_l ,W_r , D_r).

6.2.5.2. Neural net model of balance scale. McClelland (1995) has shown that a three-layered net (with input, hidden and output layers of units) can be trained to indicate whether a beam will balance, given weight and distance on left and right as input. The model accounts for a number of important empirical observations which challenge earlier theories of children's balance scale performance. However this model does not fully represent the principle of the balance scale, and the particular way it differs from the models in Section 4 is instructive. The fundamental difference is that McClelland's model does not incorporate the omni-directional access property (see 2.2.6 and 4.2.6). Weight and distance on left and right must always be inputs, and only one output, balance/left-down/right-down can be calculated. However an implementation of a quaternary relation that met the specifications in 2.2.6 can take *any* subset of N-1 variables as input and generate the Nth variable as output. For example, given weight and distance on the left, distance on the right, and the fact that the beam is balanced, it can decide what weight must be on the right. This is realistic because such tests are used in assessment (e.g., Surber & Gzesh, 1984), and because we would be unwilling to attribute understanding to a child who could compute only one type of output (such as whether the beam would balance), but could say nothing about (for example) which weights and distances were required to produce a given state of balance or imbalance. Thus while McClelland's three-layered net efficiently computes a specific function of four variables, it does not meet the criteria for relational knowledge.

6.3 Capacity development redefined

The question whether processing capacity changes with age can be reformulated by proposing that relational complexity of representations would increase because representations become differentiated into more vectors, with appropriate reconnection, as noted in 4.2.4. This would not necessarily change the total amount of information that can be processed, but it would increase the complexity of the relations that could be represented.

6.3.1 Predictions in advance. It has not been common practice for information processing theorists to publish predictions of developmental performance prior to obtaining data. However if the conceptual basis for capacity limitations advanced in this paper is more objective than previous proposals it should be possible to do this. Halford (1993, Chapter 9) made a number of such predictions. One was that two- and three-year olds should be able to make balance scale judgments based on weight or distance, but not both. The reason is that comparison of weights (or distances) on the two sides of the balance entails a binary relation, and norms indicate this is possible at age two (see 6.2.3). By contrast, Siegler (1981) found no evidence of weight or distance rules before approximately age five, and Case (1985; 1992) predicts children will not be able to judge which side will go down until three and a half to four years. Weight is likely to be easier initially because children have more experience with the downward force of weights. However with appropriate experience children should be able to make either weight or distance comparisons. This prediction was confirmed by Halford and Dalton (1995).

A further prediction was that taking account of weight and distance requires integration of binary relations, which is equivalent to at least a ternary relation, depending on the basis of the integration (see Halford, 1993, pp. 413-422 for details of the prediction). Therefore ability to consider both weight and distance in a single judgment should develop at a median age of five years, and should be predicted by performance on other measures of ternary relations. This was tested by Harper (1996) using three tasks that require ternary relations processing, but are from different domains than balance (cardinality, class inclusion, transitivity). Both predictions were confirmed.

6.3.2 Capacity and cognitive development. Halford (1993) has suggested that the observations which gave rise to cognitive developmental stage theory (Piaget, 1950) might be attributable to progressive differentiation of representations with age. In very broad terms, Piaget's secondary circular reactions, with minimal representation, correspond to single vector representations, preconceptual reasoning corresponds to unary relations (a binding of two vectors), intuitive reasoning to binary relations (a binding of three vectors), concrete operational reasoning to ternary relations (a binding of four vectors), and formal operational reasoning to quaternary relations (a binding of five vectors).

Tasks which have been considered to belong to a particular stage tend to have a common level of relational complexity. For example, transitivity and class-inclusion, which are considered to be concrete operational, entail ternary relations, as noted in 6.2.4. Each increase in complexity of relations that can be processed in parallel would enable a new level of tasks to be processed. However the fact that the theory can explain some phenomena that have been attributed to stages does not mean that all aspects of stage theory are automatically entailed. Because there is considerable potential for misinterpretation on this point we will amplify implications for cognitive development theory.

It is a common assumption that cognitive development theories that have a role for capacity *ipso facto* have no role for experience. However nothing in this theory implies that attainment of a given level of processing capacity automatically furnishes the mind with all concepts at that level. Defining a role for capacity in no way diminishes the importance of learning, induction, categorization and other acquisition processes. We have proposed that development depends on the interaction of processing capacity and acquisition processes, so that what is acquired depends both on experience and capacity (Halford, 1971; Halford, 1980; Halford & Fullerton, 1970; Halford et al., 1995; Halford & Wilson, 1980). Acquisition of transitive inference, for example, depends on experience with relations, but children who can process ternary relations will develop strategies that are more powerful and comprehensive than children who are restricted to processing one binary relation at a time. This effect has been simulated in the model of Halford et al., (1995).

Another common misconception is that capacity theories emphasize what children cannot do, and imply insurmountable barriers to performance. On the contrary, good complexity analyses can actually lead to previously unrecognized capabilities, as our work on the balance scale in 6.3.1 illustrates. Also, the discussion of chunking and segmentation in Section 3.4 shows that capacity limitations do not constitute barriers to performance. Ultimately, capacity theories are about processes rather than barriers. Saying that a particular group of participants process relations of a given complexity in parallel implies that the task must be chunked or segmented to keep within this capacity. It therefore leads to predictions about the kinds of strategies that must be employed. Thus saying that 3-year olds process binary relations in parallel implies that they will employ different strategies from 6-year olds who process ternary relations in parallel. It does not imply that they can never process transitive inferences, or other ternary relations tasks, any more than evidence that adults process quaternary relations in parallel implies we can never understand force, which ultimately depends on more than four dimensions (as discussed in 3.4.1). Capacity limitations only imply inability to perform when chunking and segmentation are inhibited, as with centre-embedded sentences in 6.1.4. Capacity theory points the way to questions that need to be investigated: For example, what chunking and serial processing abilities do children of a given age and background have in a particular domain, and how does this influence their performance? We can investigate new questions, and reexamine old questions, with this orientation.

The influence of relational complexity on cognitive development does not imply that development is discontinuous, and it is important that relational complexity theory should not be confused with traditional stage theory in this respect. Processing capacity is an *enabling factor*, but development is experience-driven and continuous, so acquisition of concepts at a given level will occur gradually after capacity becomes sufficient. There is an unlimited number of concepts belonging to a given level of complexity and acquisition of each will be a function of experience in the relevant domain, with what is learned being influenced by capacity. Furthermore acquisition of less complex concepts does not cease once capacity increases to a higher level of dimensionality, because there is an unlimited number of concepts at all levels, so when a child becomes capable of processing (say) ternary relations she does not cease acquiring concepts based on unary or binary relations.

The ages at which each level of relational complexity is typically attained should be seen as medians, with the proportion of children who attain a given level increasing gradually, in accordance with a biological growth function. The specific ages are determined empirically, and are essentially normative. Thus if it could be shown that (say) three year olds could represent ternary relations this would revise the age norms, but would not in itself invalidate the theory.

However there should be correspondence between attainment of different concepts of the same level of complexity, provided domain knowledge is adequate. For example transitivity, class inclusion and other concepts requiring ternary relations, and which there is plenty of opportunity to learn, should have the same acquisition function, a prediction confirmed by

Andrews (1996). Capacity to process relations of a given level of complexity should predict ability to acquire concepts at that level. Thus training asymptotes for concepts at a given level of complexity are the best data for testing the theory, and this methodology has been used by, for example, Halford (1980) and Halford & Leitch (1989).

The theory would be invalidated if it were found that relational complexity did not predict processing demand. However this can be tested in many ways, including some that are not developmental. Research discussed in 6.1 illustrates how relational complexity can be manipulated precisely, with other factors controlled, resulting in clearcut effects on processing demand as indicated by objective indicators such as concurrent probe reaction time. There is potential to use the same methodology with brain imaging techniques.

The cognitive developmental aspect of relational complexity grew out of levels of representational structure defined by Halford and Wilson (1980), and is consistent with neo-Piagetian theories (Case, 1985; 1992; Chapman, 1987; Pascual-Leone, 1970). There is consensus here that growth of processing capacity is an enabling factor that has an explanatory role in cognitive development, with the important *caveat* that it is not the only factor, as noted above. The relational complexity metric has potential to refine theories of processing capacity by providing a clearer mathematical definition, which should facilitate objective task analyses. It also opens up possibilities for computational modeling of growth in capacity, by differentiating neural nets into more dimensions, with consequent increases in the complexity of interactions.

Although no other cognitive developmental theory has used the metric proposed here, there is a broad parallel to the major stages defined by Case (1985, 1992), Fischer (Fischer, 1980) and Piaget (1950), the M-space levels defined by Pascual-Leone (1970), and the number of representational schemes defined by Chapman (1987) but there are also many differences. We will focus on the theory of Case (1985, 1992) who proposes that cognitive development progresses through four major stages, the sensorimotor, relational, dimensional and vectorial. There are four substages, operational consolidation (preliminary), operational coordination, bifocal coordination and elaborated coordination, which recur in each major stage. The executive processing load (equivalent to demand) is defined as OP + *S*. OP is specific to the major stage, so there is $\text{OP}_{\text{sensorimotor}}$, $\text{OP}_{\text{relational}}$, $\text{OP}_{\text{dimensional}}$ and $\text{OP}_{\text{vectorial}}$, but no quantitative value is specified for OP. One role of the present theory is to fill that gap.

The substages are quantified by the demands they make on short term memory:

. ". . the number of goals children can maintain (and hence the complexity of problem they can solve) is determined by the size of their short-term memory for the particular class of operations in question. . . this Short-Term Storage space (STSS) can hold 1,2,3, and 4 items at the preliminary, first, second, and third substage of each period, respectively." (Case, 1992, p. 32).

Notice that processing capacity, defined in terms of number of goals, ultimately depends on short-term *storage* capacity, rather than being defined in terms of information that is being processed, as in the relational complexity metric. More importantly, this progression described by Case (1985, 1992) occurs recursively in each of the major stages, so S is maximal when the highest substage is reached, and minimal when transition is made to the next major stage. This means that a task which imposes an executive processing load of 4 when performed at (say) the relational stage imposes a load of 1 when performed at the dimensional stage. The metric applies to progression within each major stage, but does not transcend major stages, so it is not possible to compare (say) the operational coordination substage of the relational stage, and the bifocal coordination substage of the vectorial stage, using a common metric (the values would be 2 and 3 respectively, but they are on different scales). Task demands are therefore assessed according to the major stages to which they are considered to belong.

However the relational complexity metric proposed here is not stage dependent in this way. It applies throughout the age range, and is also applicable to nonhuman primates, to be discussed in 6.4. If participants of any age perform the same task, encoded in the same way and using the same strategy, task demand is the same. Processing demand may well change as expertise is acquired, because rules may be discovered which simplify decision making. For example, demand may be reduced in the TOH once participants realise that the difficult decisions need only be made on the first and every fourth step thereafter, but this can be taken into account in the analysis of processing demands. Demand can vary with coding (e.g., if dimensions are chunked) or as a function of strategy (e.g., if the task is segmented into more steps requiring less information to be processed in parallel), but these factors are not stage dependent. A well-validated model of task performance enables processing demand to be analysed objectively without assignment to substages.

Case (1985) also proposes that total processing space is constant over age, and is flexibly allocated to operating space plus short term storage space, but the expected tradeoff between processing and storage does not occur (Halford et al., 1994). Perhaps the most important difference between the present theory and that of both Case (1985) and Pascual-Leone (1970) is that in the present theory processing capacity is not just a matter of space availability, but is linked to the way vectors representing dimensions of a task are connected together. As Figures 1 and 2 illustrate, processing capacity ultimately depends on connections between related entities.

This formulation also differs from Piagetian and neo-Piagetian theories in that it does not postulate substages. This is not an oversight, but is a natural consequence of the way the theory is formulated. As noted above, development is continuous, and depends both on growth of processing capacity and on acquisition processes. Substages may be used for descriptive convenience, but they are no more necessary to account for development than to account for (say) acquisition of expertise in adulthood. This should not be misinterpreted to mean that performance does not change over short periods of time. The way a task such as the Tower of Hanoi is performed may well change radically, even over a few trials, because of changes in encoding of sub-problems (e.g., chunking into pyramids as noted in 6.1.3). Acquisition mechanisms, including learning, induction, and categorization can operate over a short or a long time-frame, and are only modified by processing capacity insofar as it operates as an enabling factor.

6.4 Relational complexity and other primates

Though there is little doubt that non-human mammals have representations (Gallistel, 1990) only the primates appear to be able to process explicit relational representations that meet the criteria in 2.2. We will briefly review this evidence.

Premack (1983) reports that tasks which require symbolic representations differentiate chimpanzees and monkeys from lower mammals, whereas tasks based on perceptible similarity, or on inferences about spatial location, do not. We propose that relational representations subsume symbolic processes, because of the properties of relational knowledge in 2.2. One of Premack's procedures requires chimpanzees (*Pan troglodytes*) to identify the relation, given a pair of arguments, or to identify an argument, given the relation and the other argument (omni-directional access as defined in 2.2.6). In the former case, for example, they would be shown two objects and asked to produce a symbol indicating whether the objects were the same or different. They also seem to know that a knife corresponds to the relation between an intact and a cut object, a key to the relation between a closed and an open lock, and so on. These tasks seem to require recognition of a symbol for a binary relation; a knife is a symbol for the relation between an intact and a cut object (the knife functions as a relation symbol, see 2.2.4 and 4.2.3).

Another task used by Premack has been analysed in terms of relational complexity by Holyoak and Thagard (1995). Chimpanzees were required to choose a pair of objects which had the same relation as a sample pair. We will call it "relational-match-to-sample task". In one variant, the sample comprised two objects that were the same (XX). The participant had to choose another pair of objects that were the same (YY), in preference to a pair that were different (CD). The task is a form of analogical reasoning, as Premack (1983) points out, and requires representation of the relation between elements in the pair. The sample is the base, and the comparison pair is the target. Following Holyoak and Thagard (1995) we can code the sample as O-same(X,X), where "O-same" means "same object". The correct comparison (target) object is represented as O-same(Y,Y). In the alternate task, the base would be coded as O-different(X,Y), and the correct target as O-different(C,D). Only chimpanzees, and only those that had been language trained, could perform this task. It seems reasonable to conclude that chimpanzees can process binary relations, albeit only after extensive experience with symbols. However there appears to be no evidence that analogies based on binary relations can be attributed to lower animals.

The one-object match-to-sample entails presenting a sample, X, and rewarding animals for choosing the comparison object that is the same as the sample, X, in preference to the one that is different, Y. Chimpanzees and some monkeys can generalize the performance beyond instantiations on which they have been trained. This requires responding to an attribute or a category label, which can be coded as a unary relation (as shown in 2.3.1). For example, if the sample were an apple, this can be represented as apple($object_1$). If the comparison objects were an apple and an orange, they would be represented, as apple($object_2$), orange($object_3$). The one-object match-to-sample task is an analogy based on a unary relation: it is a mapping from apple($object_1$) to apple($object_2$). Thus some species of monkeys appear to process unary relations.

6.5 Role of frontal lobes

In a review of the literature, Robin and Holyoak (1995) propose that the prefrontal cortex functions as an overall system for constructing and maintaining relational representations that guide thought and action. They argue that many of the functions that are impaired with frontal lobe lesions, including planning and control, entail 2- and 3-dimensional representations, with dynamic binding. According to this hypothesis the species- and age-differentiations discussed earlier would be attributed to the observation that the frontal lobes evolve later and are slower to myelinate. This formulation, together with associated empirical work, enables relational complexity of tasks to be manipulated with other factors controlled. These techniques have reached high levels of refinement (e.g., Andrews, 1996) and are now ready to be used with brain imaging techniques. It would be predicted that tasks which require more complex relations to be processed in parallel would produce more activation in the prefrontal cortex, with other factors controlled.

7.0 Conclusions

The empirical data base in cognitive psychology, and current neural net models of relational knowledge, indicate that processing capacity is not limited by amount of information or number of items per se, but by the number of independent dimensions that can be related in parallel. Relational complexity, defined as the number of independent sources of variation that are related, constitutes a major factor underlying the difficulty of higher cognitive processes. It is related to processing load, to differences between higher animal species, and to age in children. There is potential to explain processing loads by modeling neural net representations of relations.

The theory provides a way of blending serial and parallel processes, by proposing that serial processing is necessitated by limitations in the complexity of structures that can be processed in parallel. Empirical data and contemporary neural net models of relational knowledge indicate that the most complex structure that can be processed in parallel, and without crosstalk, is equivalent to one quaternary relation. More complex representations must either be chunked into fewer components, with the result that some of the relational structure becomes temporarily inaccessible, or the task must be segmented into smaller components that are processed serially, or both. Thus the need for serial processing strategies can be seen as a consequence of processing capacity limitations.

The theory implies that the traditional approach of defining limitations in terms of items is inappropriate for processing capacity, although some common ground is found with Miller's (1956) suggestion that the limit was defined by the number of independent components, rather than the amount of information. The concept of a chunk has been retained, but has been extended to include conceptual chunks, which represent compressed relational instances. The definition of capacity in terms of relational complexity, and the exploration of possible neural net implementations, has given a new way of looking at many issues, and one which integrates a wide-ranging data base. In cognitive development it means that the issue is no longer simply whether capacity changes with age, but whether representations become differentiated into higher dimensionalities, so more complex relations can be processed.

References

Anderson, J. R., Reder, L. M., & Lebiere, C. (1996). Working memory: Activation limitations on retrieval. Cognitive Psychology, 30 (3), 221-256.

Anderson, M. (1992). Intelligence and development: A cognitive theory. Oxford UK Cambridge USA: Blackwell.

Andrews, G. (1996, August). Assessment of relational reasoning in children aged 4 to 8 years. Paper presented at the Conference of the International Society for Study of Behavioral Development, Quebec City.

Andrews, G., & Halford, G. S. (1994,). Relational Complexity and Sentence Processing. Proceedings of the 21st Annual Experimental Psychology Conference, University of Sydney, 24.

Andrews, G., & Halford, G. S. (submitted). Children's ability to make transitive inferences: The importance of analogical mapping and structural complexity.

Astington, J. W. (1993). The Child's Discovery of Mind. Cambridge, MA: Harvard University Press.

Attneave, F. (1959). Applications of information theory to psychology: A summary of basic concepts, methods and results. New York: Henry Holt & Company.

Baddeley, A., & Hitch, G. (1974). Working memory. In G. H. Bower (Ed.), The psychology of learning and motivation: Advances in research and theory (Vol. 8, pp. 47-89). New York: Academic Press.

Baddeley, A. D. (1986). Working Memory. Oxford: Clarendon Press.

Baddeley, A. D. (1990). Human memory: Theory and practice. Needham Heights, MA: Allyn & Bacon.

Baillargeon, R. (1987a). Object permanence in 3 1/2- and 4 1/2-month-old infants. Developmental Psychology, 23, 655-664.

Baillargeon, R. (1987b). Young infants' reasoning about the physical and spatial properties of a hidden object. Cognitive Development, 2, 179-200.

Bitterman, M. E. (1960). Toward a comparative psychology of learning. American Psychologist, 15, 704-712.

Bitterman, M. E. (1975). The comparative analysis of learning. Science, 188, 699-709.

Breslow, L. (1981). Reevaluation of the literature on the development of transitive inferences. Psychological Bulletin, 89, 325-351.

Broadbent, D. E. (1975). The magic number seven after fifteen years. London: Wiley.

Bryant, P. E., & Trabasso, T. (1971). Transitive inferences and memory in young children. Nature, 232, 456-458.

Carpenter, P. A., & Just, M. A. (1996). Functional and neuroscience approaches to working memory. International Journal of Psychology, 31(3/4), 325.

Case, R. (1985). Intellectual development: Birth to adulthood. New York: Academic Press.

Case, R. (1992). The mind's staircase: Exploring the conceptual underpinnings of children's thought and knowledge. Hillsdale, NJ: Erlbaum.

Chapman, M. (1987). Piaget, attentional capacity, and the functional limitations of formal structure. Advances in Child Development and Behaviour, 20, 289-334.

Clark, A., & Karmiloff-Smith, A. (1993). The cognizer's innards: A psychological and philosophical perspective on the development of thought. Mind and Language, 8(4), 487-519.

Codd, E. F. (1990). The Relational Model for Database Management: Version 2: Addison-Wesley.

Cummins, D. D., Kintsch, W., Reussler, K., & Weimer, R. (1988). The role of understanding in solving word problems. Cognitive Psychology, 20(4), 405-438.

Davis, H. L., & Pratt, C. (1995). The development of children's theory of mind: The working memory explanation. Australian Journal of Psychology, 47(1), 25-31.

English, L. D., & Halford, G. S. (1995). Mathematics education: Models and processes. Hillsdale, NJ: Erlbaum.

Fischer, K. W. (1980). A theory of cognitive development: The control and construction of hierarchies of skills. Psychological Review, 87, 477-531.

Fisher, D. L. (1984). Central capacity limits in consistent mapping, visual search tasks: Four channels or more? Cognitive Psychology, 16(4), 449-484.

Flavell, J. H., Green, F. L., & Flavell, E. R. (1990). Developmental changes in young children's knowledge about the mind. Cognitive Development, 5(1), 1-27.

Fodor, J. A. (1983). Modularity of mind: An essay on faculty psychology. Cambridge, MA: MIT Press.

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

Frye, D., Zelazo, P. D., & Palfai, T. (1995). Theory of mind and rule-based reasoning. Cognitive Development, 10, 483-527.

Gallistel, C. R. (1990). Representations in animal cognition: an introduction. Cognition, 37, 1-22.

Garey, M. R., & Johnson, D. S. (1979). Computers and intractability: A guide to the theory of NP-completeness. San Francisco: W. H. Freeman.

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

Gick, M. L., & Holyoak, K. J. (1983). Schema induction and analogical transfer. Cognitive Psychology, 15, 1-38.

Gopher, D. (1994). Analysis and measurement of mental load. In G. d'Ydewalle, P. Eelen, & P. Bertelson (Eds.), The state of the art (Vol. 2, pp. 265-291). Hove, England: Lawrence Erlbaum Associates.

Goswami, U. (1992). Analogical reasoning in children. Hove, England; Hillsdale, NJ: Laurence Erlbaum Associates.

Goswami, U. (1995). Transitive relational mappings in 3- and 4-year-olds: The analogy of Goldilocks and the three bears. Child Development, 66, 877-892.

Greeno, J. G., Riley, M. S., & Gelman, R. (1984). Conceptual competence and children's counting. Cognitive Psychology, 16, 94-143.

Grieve, R., & Garton, A. (1981). On the young child's comparison of sets. Journal of Experimental Child Psychology, 32, 443-458.

Halford, G. S. (1971). Acquisition of conservation through learning a consistent classificatory system for quantities. Australian Journal of Psychology, 23, 151-159.

Halford, G. S. (1980). A learning set approach to multiple classification: Evidence for a theory of cognitive levels. International Journal of Behavioral Development, 3, 409-422.

Halford, G. S. (1982). The development of thought. Hillsdale, NJ: Lawrence Erlbaum Associates.

Halford, G. S. (1984). Can young children integrate premises in transitivity and serial order tasks? Cognitive Psychology, 16, 65-93.

Halford, G. S. (1989). Reflections on 25 years of Piagetian cognitive developmental psychology, 1963-1988. Human Development, 32, 325-387.

Halford, G. S. (1992). Analogical reasoning and conceptual complexity in cognitive development. Human Development, 35, 193-217.

Halford, G. S. (1993). Children's understanding: the development of mental models. Hillsdale, N. J.: Erlbaum.

Halford, G. S. (1996). Children's understanding of the mind: An instance of a general principle? [Review of The Child's Discovery of the Mind]. Contemporary Psychology, 41, 229-230.

Halford, G. S., Bain, J. D., & Maybery, M. T. (1984). Does a concurrent memory load interfere with reasoning? Current Psychological Research and Reviews, 3, 14-23.

Halford, G. S., & Dalton, C. (1995). Performance on the balance scale by 2-year-old children. (ERIC Document Reproduction Service No. ED 385 355)

Halford, G. S., & Fullerton, T. (1970). A discrimination task which induces conservation of number. Child Development, 41, 205-213.

Halford, G. S., & Kelly, M. E. (1984). On the basis of early transitivity judgements. Journal of Experimental Child Psychology, 38, 42-63.

Halford, G. S., & Leitch, E. (1989). Processing load constraints: A structure-mapping approach. In M. A. Luszcz & T. Nettelbeck (Eds.), Psychological development: Perspectives across the life-span (pp. 151-159). Amsterdam: North-Holland.

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., Maybery, M. T., & Bain, J. D. (1988). Set-size effects in primary memory: An age-related capacity limitation? Memory and Cognition, 16(5), 480-487.

Halford, G. S., Maybery, M. T., O'Hare, A. W., & Grant, P. (1994). The development of memory and processing capacity. Child Develpoment, 65, 1338-1356.

Halford, G. S., Smith, S. B., Dickson, J. C., Maybery, M. T., Kelly, M. E., Bain, J. D., & Stewart, J. E. M. (1995). Modelling the development of reasoning strategies: The roles of analogy, knowledge, and capacity. In T. Simon & G. S. Halford (Eds.), Developing Cognitive Competence: New Approaches to Cognitive Modelling . Hillsdale, NJ: Erlbaum.

Halford, G. S., & Wilson, W. H. (1980). A category theory approach to cognitive development. Cognitive Psychology, 12, 356-411.

Halford, G. S., Wilson, W. H., Guo, J., Gayler, R. W., Wiles, J., & Stewart, J. E. M. (1994). Connectionist implications for processing capacity limitations in analogies. In K. J. Holyoak & J. Barnden (Eds.), Advances in connnectionist and neural computation theory, Vol. 2: Analogical connections (pp. 363-415). Norwood, NJ: Ablex.

Halford, G. S., Wilson, W. H., & McDonald, M. (1995, July). Complexity of structure mapping in human analogical reasoning: A PDP model. Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, Pittsburgh, Pennsylvania, 597-601.

Halford, G. S., Wilson, W. H., & Phillips, S. (1996). Human analogical reasoning capacity: Toward a neural net model. International Journal of Psychology, 31(3/4), 443.

Harper, M. (1996). Complexity as a factor in children's performance on the balance scale. Unpublished Honours Thesis, University of Queensland, Brisbane, Australia.

Henderson, J. (1994). Connectionist syntactic parsing using temporal variable binding. Journal of Psycholinguistic Research, 23(5), 353-379.

Hitch, G. J. (1980). Developing the concept of working memory. London: Routledge & Kegan Paul.

Hodkin, B. (1987). Performance model analysis in class inclusion: An illustration with two language conditions. Developmental Psychology, 23, 683-689.

Holland, J. H., Holyoak, K. J., Nisbett, R. E., & Thagard, P. R. (1986). Induction: Processes of inference, learning and discovery. Cambridge, MA: Bradford Books/MIT Press.

Holyoak, K. J., & Thagard, P. (1989). Analogical mapping by constraint satisfaction. Cognitive Science, 13(3), 295-355.

Holyoak, K. J., & Thagard, P. (1995). Mental leaps. Cambridge, MA: MIT Press.

Hummel, J. E., & Holyoak, K. J. (in press). Distributed representations of structure: A theory of analogical access and mapping. *Psychological Review*.

Humphreys, M. S., Bain, J. D., & Pike, R. (1989). Different ways to cue a coherent memory system: A theory for episodic, semantic and procedural tasks. Psychological Review, 96(2), 208-233.

James, W. (1890). Principles of psychology. New York: Holt, Rinehart & Winston.

Johnson-Laird, P. N., Byrne, R. M. J., & Schaeken, W. (1992). Propositional reasoning by model. Psychological Review, 99, 418-439.

Just, M. A., & Carpenter, P. A. (1992). A capacity theory of comprehension: Individual differences in working memory. Psychological Review, 99(1), 122-149.

Just, M. A., Carpenter, P. A., & Hemphill, D. D. (in press). Constraints on processing capacity: Architectural or implementational? In D. Steier & T. Mitchell (Eds.), Mind Matters: A Tribute to Allen Newell .

Kahneman, D. (1973). Attention and effort. Englewood Cliffs, NJ: Prentice-Hall.

Kallio, K. D. (1982). Developmental change on a five-term transitive inference. Journal of Experimental Child Psychology, 33, 142-164.

Kimball, J. (1973). Seven principles of surface structure parsing in natural language. Cognition, 2, 15-47.

Kintsch, W., & Greeno, J. G. (1985). Understanding and solving word arithmetic problems. Psychological Review, 92(1), 109-129.

Klahr, D., & Wallace, J. G. (1976). Cognitive development: An information processing view. Hillsdale, NJ: Lawrence Erlbaum Associates.

Klapp, S. T., Marshburn, E. A., & Lester, P. T. (1983). Short-term memory does not involve the "working memory" of information processing: The demise of a common assumption. Journal of Experimental Psychology: General, 112, 240-264.

Logan, G. D. (1979). On the use of a concurrent memory load to measure attention and automaticity. Journal of Experimental Psychology: Human Perception and Performance, 5, 189-207.

Loveday. (1995). The effect of complexity on planning in the Tower of Hanoi problem. Unpublished Honours Thesis, University of Queensland.

Markman, E. M., & Seibert, J. (1976). Classes and collections: Internal organization and resulting holistic properties. Cognitive Psychology, 8, 561-577.

Maybery, M. T., Bain, J. D., & Halford, G. S. (1986). Information processing demands of transitive inference. Journal of Experimental Psychology: Learning, Memory and Cognition, 12, 600-613.

McClelland, J. L. (1995). A connectionist perspective on knowledge and development. In T. Simon & G. S. Halford (Eds.), Developing Cognitive Competence: New Approaches to Cognitive Modelling (pp. 157-204). Hillsdale, NJ: Erlbaum.

McGarrigle, J., Grieve, R., & Hughes, M. (1978). Interpreting inclusion: A contribution to the study of the child's cognitive and linguistic development. Journal of Experimental Child Psychology, 26, 528-550.

Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. Psychological Review, 63, 81-97.

Navon, D., & Gopher, D. (1980). Task difficulty, resources and dual task performance. Hillsdale, NJ: Erlbaum.

Pascual-Leone, J. A. (1970). A mathematical model for the transition rule in Piaget's developmental stages. Acta Psychologica, 32, 301-345.

Pears, R., & Bryant, P. (1990). Transitive inferences by young children about spatial position. British Journal of Psychology, 81(4), 497-510.

Phillips, S., Halford, G. S., & Wilson, W. H. (1995). The processing of associations versus the processing of relations and symbols: A systematic comparison. Proceedings of the Seventeenth Annual Conference of the Cognitive Science Society, Pittsburgh,PA, 688-691.

Phillips, S., Halford, G. S., & Wilson, W. H. (submitted). Representational redescription: From associative to relational systems.

Piaget, J. (1950). The psychology of intelligence. (M. Piercy & D. E. Berlyne, Trans.) London: Routledge & Kegan Paul, (Original work published 1947).

Plate, T. A. (1994). Distributed representations and nested compositional structure. Unpublished PhD Thesis, University of Toronto, Toronto, Canada.

Plate, T. A. (1995). Holographic reduced representations. IEEE Transactions on Neural Networks, 6(3), 623-641.

Plate, T. A. (in press). Estimating analogical similarity by vector dot-products of holographic reduced representations. Cognitive Science.

Posner, M. I., & Boies, S. J. (1971). Components of attention. Psychological Review, 78, 391-408.

Premack, D. (1983). The codes of man and beasts. The Behavioral and Brain Sciences, 6, 125-167.

Robin, N., & Holyoak, K. J. (1995). Relational complexity and the functions of prefrontal cortex. In M. S. Gazzaniga (Ed.), The Cognitive Neurosciences (pp. 987-997). Cambridge, MA: MIT Press.

Schneider, W., & Detweiler, M. (1987). A connectionist/control architecture for working memory. The Psychology of Learning and Motivation, 21, 53-119.

Schweickert, R., & Boruff, B. (1986). Short-term memory capacity: Magic number or magic spell? Journal of Experimental Psychology: Learning, Memory and Cognition, 12, 419-425.

Shastri, L., & Ajjanagadde, V. (1993a). From simple associations to systematic reasoning: A connectionist representation of rules, variables, and dynamic bindings using temporal synchrony. Behavioral and Brain Sciences, 16(3), 417-494.

Shastri, L., & Ajjanagadde, V. (1993b). A step toward modeling reflexive reasoning. Behavioural and Brain Sciences, 16(3), 477-488.

Shipley, E. F. (1979). The class-inclusion task: Question form and distributive comparisons. Journal of Psycholinguistic Research, 8, 301-331.

Siegler, R. S. (1981). Developmental sequences within and between concepts. Monographs of the Society for Research in Child Development, 46, 1-84.

Smith, L. B. (1989). From global similarities to kinds of similarities: The construction of dimensions in development. In S. Vosniadou & A. Ortony (Eds.), Similarity and analogical reasoning (pp. 146-178). Cambridge, MA: Cambridge University Press.

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

Sternberg, R. J. (1980). Representation and process in linear syllogistic reasoning. Journal of Experimental Psychology: General, 109, 119-159.

Sugarman, S. (1982). Developmental change in early representational intelligence: Evidence from spatial classification strategies and related verbal expressions. Cognitive Psychology, 14, 410-449.

Surber, C. F., & Gzesh, S. M. (1984). Reversible operations in the balance scale task. Journal of Experimental Child Psychology, 38, 254-274.

Sweller. (1993). Some cognitive processes and their consequences for the organisation and presentation of information. Australian Journal of Psychology, 45(1), 1-8.

Tesar, B. B., & Smolensky, P. (1994,). Synchronous firing variable binding is a tensor product representation with temporal role vectors. 16th Annual Conference of the Cognitive Science Society, Atlanta, Georgia, 870-875.

Thayer, E. S., & Collyer, C. E. (1978). The development of transitive inference: A review of recent approaches. Psychological Bulletin, 85, 1327-1343.

Trabasso, T. (1975). Representation, memory, and reasoning: How do we make transitive inferences? Minneapolis: University of Minnesota Press.

Trabasso, T. (1977). The role of memory as a system in making transitive inferences. In R. V. Kail & J. W. Hagen (Eds.), Perspectives on the development of memory and cognition (pp. 333-366). Hillsdale, NJ: Lawrence Erlbaum Associates.

Trabasso, T., Isen, A. M., Dolecki, P., McLanahan, A. G., Riley, C. A., & Tucker, T. (1978). How do children solve class-inclusion problems? In R. Siegler (Ed.), Children's thinking: What develops? (pp. 151-180). Hillsdale, NJ: Lawrence Erlbaum Associates.

Tsotsos, J. K. (1990). Analyzing vision at the complexity level. Behavioral and Brain Sciences, 13, 423-469.

VanLehn, K. (1991). Rule acquisition events in the discovery of problem-solving strategies. Cognitive Science, 15(1), 1-47.

VanLehn, K., & Brown, J. S. (1980). Planning nets: A representation for formalizing analogies and semantic models of procedural skills. In R. E. Snow, P. A. Federico, & W. E. Montague (Eds.), Aptitude learning and instruction. Vol. 2. Cognitive process analyses of learning and problem solving (pp. 95-137). Hillsdale, NJ: Lawrence Erlbaum Associates.

Wellman, H. M., Cross, D., & Bartsch, K. (1986). Infant search and object permanence: A meta-analysis of the A-not-B error. Monographs of the Society for Research in Child Development, 51.

Wilkinson, A. (1976). Counting strategies and semantic analyses as applied to class inclusion. Cognitive Psychology, 8, 64-85.

Wilson, W. H., & Halford, G. S. (1994). Robustness of tensor product networks using distributed representations. Fifth Australian Conference on Neural Networks, Brisbane, 258-261.

Wilson, W. H., Street, D. J., & Halford, G. S. (1995). Solving proportional analogy problems using tensor product networks with random representations. IEEE International Conference on Neural Networks Proceedings, Perth, Australia, 2971-2975.

Wimmer, H., & Perner, J. (1983). Beliefs about beliefs: Representation and constraining function of wrong beliefs in young children's understanding of deception. Cognition, 13, 103-128.

Younger, B. (1993). Understanding category members as "the same sort of thing": Explicit categorization in ten month infants. Child Development, 64(1), 309-320.

Yuille, A. L., & Geiger, D. (1995). Winner-take-all mechanisms. In M. A. Arbib (Ed.), The handbook of brain-theory and neural networks (pp. 1056-1060). Cambridge, MA: MIT Press.

Appendix A: Relations derived by decomposing higher-rank relations

We will consider the example MONOTONICALLY-LARGER(a,b,c) discussed in 2.2.9. Let us abbreviate MONOTONICALLY-LARGER to ML. Then it is interesting to look at the derived relations (projection relations) ML_1, ML_2, and ML_3. $ML_1(b,c)$ means that $b > c$ and there is some item x such that $x > b$. Similarly, $ML_2(a,c)$ means that $a > c$ and there is some item x such that $a > x > c$. Thus each ML_i is a sub-relation of the relation > (GREATER-THAN). Thus the ternary relation induces a number of binary relations, and in this case ML(a,b,c) can be reconstructed from the induced relations in the sense that $ML(a,b,c) \equiv ML_1(b,c)$ & $ML_2(a,c)$ & $ML_3(a,b)$. By contrast, the ternary relation $R(x,y,z) \equiv x > yz$, where x,y, and z are positive rational numbers, is not reconstructable, since each of the induced relations is the "trivial relation" - e.g., for each pair of numbers y and z, there exists an x such that $x > yz$ (for example $yz+1$). Thus $R_1(y,z)$ is true for all y and z, and the same is true for R_2 and R_3. Thus R is not effectively decomposable, basically because of the presence of the binary operation yz.

Appendix B: Retrieval from tensor product representation of symbol-argument-argument bindings.

The outer product T of a set of three vectors **u**, **v**, **w**, is a multi-dimensional object like a generalized matrix with 3 subscripts i, j, k. The (i, j, k)-th entry in the outer product is defined to be the product of the corresponding entries in each vector: $T_{ijk} = \mathbf{u}_i\mathbf{v}_j\mathbf{w}_k$. This definition generalizes in a natural way to as many vectors as required.

Generalised Inner Products: For a tensor T of a given rank, a number of retrieval operations can be defined. For example, with a rank 4 tensor $T = T_{pqrs}$ (storing relational instances of the form $r(a_1, a_2, a_3)$, say) one may wish to:

(0) check the validity of a particular relational instance r(a,b,c);

(1) find the X's such that $r(X,a_2,a_3)$ is stored in the tensor;

(2) find the X's such that $r(a_1,X,a_3)$ is stored in the tensor;

(3) find the X's such that $r(a_1,a_2,X)$ is stored in the tensor;

(4) find the X's such that $X(a_1,a_2,a_3)$ is stored in the tensor;

Other retrievals are possible, such as finding the pairs X,Y such that $r(X,Y,a_3)$ is stored in the tensor, but we shall restrict ourselves to modes (0)-(4) here. Each of the five operations required is a kind of generalized inner product. For convenience, let us define notation that distinguishes between the five kinds of generalized inner product:

(0) $v_r \otimes v_{a1} \otimes v_{a2} \otimes v_{a3} \bullet T$ checks validity;

(1) $v_r \otimes \_ \otimes v_{a2} \otimes v_{a3} \bullet T$ retrieves a_1's;

(2) $v_r \otimes v_{a1} \otimes \_ \otimes v_{a3} \bullet T$ retrieves a_2's;

(3) $v_r \otimes v_{a1} \otimes v_{a2} \otimes \_ \bullet T$ retrieves a_3's;

(4) $\_ \otimes v_{a1} \otimes v_{a2} \otimes v_{a3} \bullet T$ retrieves r's;

Now we give the computation required for each kind of retrieval. Let r_p signify the p^{th} component of the vector v_r representing r; and let $(a_1)_q$, $(a_2)_r$, and $(a_3)_s$ signify respectively the q^{th}, r^{th}, and s^{th} components of the vectors v_{a1}, v_{a2}, and v_{a3} representing a_1, a_2, and a_3.

(0) Let $D = \sum_{pqrs} r_p(a_1)_q(a_2)_r(a_3)_s T_{pqrs}$. If D = 1, then $r(a_1,a_2,a_3)$ is stored in **T**. Otherwise D=0, and $r(a_1,a_2,a_3)$ is not stored in **T**.

For (1)-(4), let **v** be the vector representing the missing concept X, e.g. a_1 in operation (1).

(1) $v_q = \sum_{prs} r_p (a_2)_r(a_3)_s T_{pqrs}$

(2) $v_r = \sum_{pqs} r_p(a_1)_q (a_3)_s T_{pqrs}$

(3) $v_s = \sum_{pqr} r_p(a_1)_q(a_2)_r T_{pqrs}$

(4) $v_p = \sum_{qrs} (a_1)_q(a_2)_r(a_3)_s T_{pqrs}$

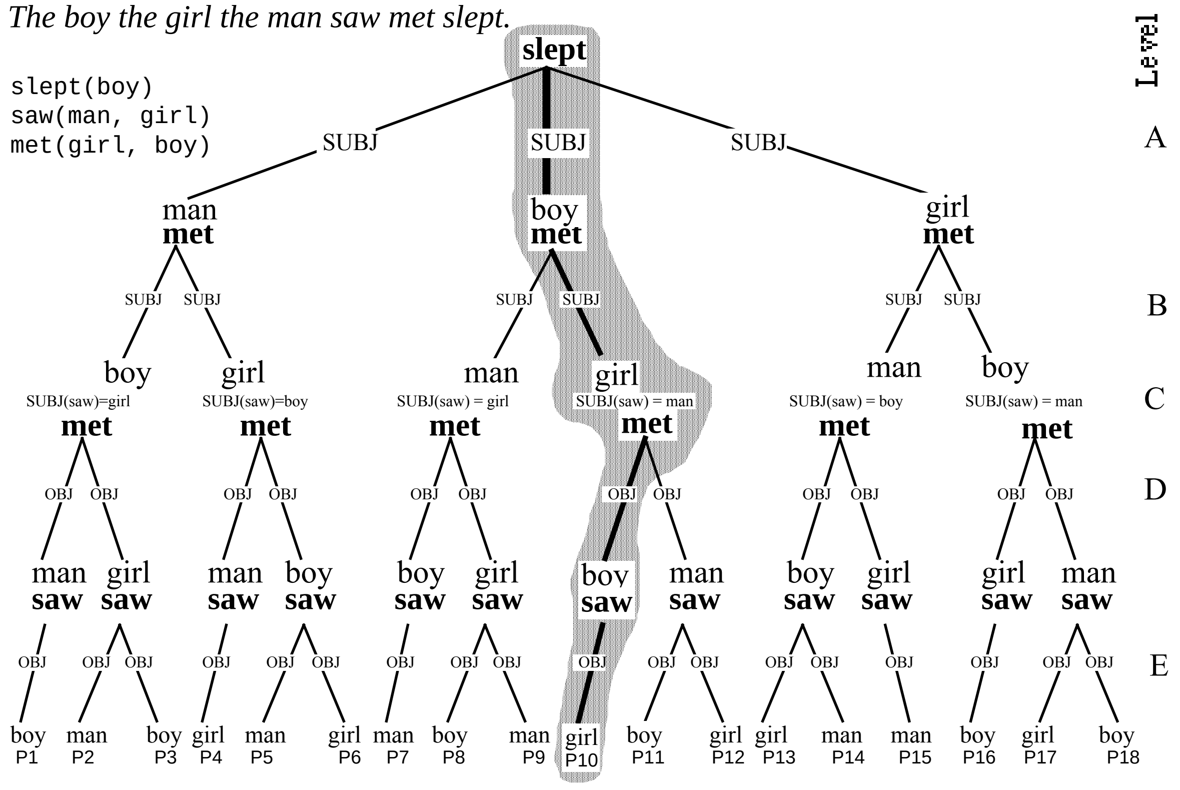
Similar operations can be specified for tensors of lower and higher ranks. For rank 2, the operations reduce to matrix pre- and/or post-multiplication by vector(s).

It may be useful to spell out the details of retrievals for a set of binary operations represented in a rank 3 tensor product space, as another example.

We assume a rank three tensor $T_{ijk} \in V_P \otimes V_1 \otimes V_2$ and consider the three possible "directions" of access: (a) we know the two arguments, represented by vectors **u** $\in V_1$ and **v** $\in V_2$, and we want to know the relation-symbol(s) **p** $\in V_P$ such that **p**$\otimes$**u**$\otimes$**v**, that is, **p**(**u**,**v**) is a fact "known to the tensor"; (b) we know **p** and **u**, and want to find out the **v** for which **p**(**u**,**v**) is known; (c) we know **p** and **v**, and want to find out the **u** for which **p**(**u**,**v**) is known. In fact, $\sum_j \sum_k T_{ijk} u_j v_k$ is the answer to (a), in the sense that this expression is a vector (subscripted by i) which is the sum of all the vectors representing symbols **p** such that **p**(**u**,**v**) is known. Similarly, $\sum_i \sum_j T_{ijk} p_i u_j$ is the answer to (b), and that $\sum_i \sum_k T_{ijk} p_i v_k$ is the answer to (c). Because the value of any argument can be computed given the values of all the other arguments, variations in any dimension as a function of the others can be computed.

Appendix C. Parsing of centre embedded sentence.

The boy the girl the man saw met slept.



Combinatorial Rules:

1 subjects of distinct verbs must be distinct

2 objects of distinct verbs must be distinct

Combinatorial/Grammatical Rule:

3 a single noun phrase cannot be both the subject and the object of the same verb

Case/Grammatical Rules:

4 NP•TV $\rightarrow$ S\NPOBJ: SUBJ(TV) = NP

5 NP_1•S\NPOBJ $\rightarrow$ NP_2: OBJ(VERB(S\NPOBJ)) = NP_1

6 TV•NP $\rightarrow$ VP: OBJ(TV) = NP; NP•VP $\rightarrow$ S

7 NP•IV $\rightarrow$ S: SUBJ(IV) = NP

In the case/grammatical rules, TV signifies Transitive Verb, and IV signifies Intransitive Verb. The rules are context-free rules written back to front, augmented with case assignments. Thus rule 5 could be read: "if you find an NP_1 (noun phrase) followed by an S\NPOBJ (sentence with omitted object NP) then you have found a (higher-level) noun phrase NP_2: set the object slot of the verb in the $S\backslash NP_{OBJ}$ to be the noun phrase NP_1.

Commentary: The tree at the top shows the *combinatorial* choices for analysis of the sentence *The boy the girl the man saw met slept*. Subject to rules 1-3, there are 18 combinatorial possibilities, labelled P1-P18.

Levels are shown at right: the 3-way split at level A shows the three-way choice of subject for the verb *slept*.

The split at level B for the subject of *met* is 2-way, since rule 1 eliminates one of the possibilities - for example, in the left most sub-tree at level B, the SUBJ cannot be *man* since *man* has already been used as the subject of *slept*.. At level C, similarly, there is only one choice in each case for the subject of *saw*.

At level D, as the subject of *met* has already been chosen, because of rule 3, there are only two remaining choices for the object of *met*.

At level E, when choosing the object of *saw*, the NP both must not have been chosen as the object of *met*, (rule 2) and must not have been chosen as the subject of *saw* (rule 3). The intersection of these possibilities is sometimes just one NP, sometimes two. For example, with P16, OBJ(*saw*) cannot be *girl* because OBJ(*met*) is *girl*, and cannot be *man* because SUBJ(*saw*) = *man*. However, with P17, P18, SUBJ(*saw*) = OBJ(*met*) = *man*, so both *girl* and *boy* are possible for OBJ(*saw*).

Now the case/grammatical rules:

Apply rule 4 to *the man saw* → SUBJ(*saw*) = *man* [Possibilities satisfying this: P10, P11, P12, P16, P17, P18].

Apply rule 5 to *the girl the man saw* → OBJ(*saw*) = *girl* [Possibilities: P10, P12, P17].

Apply rule 5 to *the boy the girl the man saw met* → OBJ(*met*) = *boy* [Possibility: P10].

Apply rule 7 to *the boy the girl the man saw met* → SUBJ(*slept*) = *boy* [Possibility: P10].

Rule 6 is not used in the example sentence above, but would be used in *The boy the girl met hit Tom*.

Endnotes

[1] A dimension A is independent of another dimension, B, if, when classifying entities according to their values on dimensions A and B, knowing the value of an entity according to dimension A does not always determine the value of the entity according to dimension B.

[2] Though their significance could not be determined.

[3] Actually the problems of instance of identification are even greater than this. Not only must each relational instance be bound to a unique context vector (i.e., the context in which the relational instance was memorized), but each vector must be orthogonal (dissimilar) so as to avoid the problem of cross-talk (role vectors bound to fillers from different relational instances). The orthogonality requirement introduces a dilemma: if we choose random dissimilar vectors, then in general relational instances are not distinguishable on the basis of their contents. On the other hand, if we generate identification vectors on the basis of contents, these vectors will no longer be orthogonal since many relational instances share the components. These problems can be overcome by defining each instance by its components and linking them together, but notice that this effectively entails adopting symbol-argument-argument bindings.

[4] In the special case, where each role is represented by a unique local basis vector (e.g., [1 0 0], etc), the tensor role-filler method also maps each relation dimension onto a separate tensor subspace. However role-filler methods may entail the psychologically unrealistic assumption that each relational instance can be uniquely identified by a context independent vector.

[5] This property can be linked to compositionality (Fodor & Pylyshyn, 1988) but it is not appropriate to develop that link in this paper.

[6] Order notation $\Theta(.)$ identifies the term in n with the largest power, so that $100n^2$; $n^2 + 10n$; and $0.001n^2$ are considered to be of the same order (i.e., $\Theta(n^2)$ - quadratic).

[7] Frye et al. (1995) refer to this as a “setting condition”.

Table 1.

Tower of Hanoi moves which require planning, showing goal(s) and dimensions.

Problem/Move	Current State	Move	New State	Goal(s)*	Dimensions	
					all	new
2 disc						
1	12,_,_	1 to B	2,1,_	2C 1B	4	4
3 disc						
1	123,_,_	1 to C	23,_,1	3C 2B 1C	6	6
5	_,12, 3	1 to A	1,2,3	2C 1A		
4 disc						
1	1234,_,_	1 to B	234,1,_	4C 3B 2C 1B	8	8
5	4, 3,12	**1 to A**	14,3,2	4C 2B 1A	6	4
9	_,123,4	1 to C	_,23,14	3C 2A 1C	6	6
13	12,_, 34	1 to B	2,1, 34	2C 1B	4	4
5 disc						
1	12345,_,_	1 to C	2345,_,1	5C 4B 3C 2B 1	8	8
5	45,12,3	**1 to A**	145,2,3	5C 4B 2C 1A	8	4
9	5,**4**,123	1 to B	5,14,23	5C 3B 2A 1B	8	6
13	125,3**4**,_	1 to C	25,34,1	5C 2B 1C	6	4
17	_,1234,5	1 to A	1,234,5	4C 3A 2C 1A	8	6
21	3,4,125	**1 to B**	3,14,25	4C 2A 1B	6	4
25	123,_,45	1 to C	23,_,145	3C 2B 1C	6	4
29	_,12,345	**1 to A**	1,2,345	2C 1A	4	2

*New goals are underlined

Table 2.

Mapping ordered triples, based on absolute size

		Experimenter's set			
		1 5 9	2 6 10	3 7 11	4 8 12
	1 5 9	1 1 1	1 1 1	.5 .5 1	0 0 1
Children's	2 6 10	1 1 1	1 1 1	1 1 1	.5 .5 1
set	3 7 11	1 .5 .5	1 1 1	1 1 1	1 1 1
	4 8 12	1 0 0	1 .5 .5	1 1 1	1 1 1

1 = absolute size gives correct answer

.5 = absolute size gives 2 answers, one of which is correct

0 = absolute size gives incorrect answer

Figure Captions

Figure 1. Neural net capable of tensor product representation of role-filler binding or unary relation (A) and as arithmetic example (B). A binary relation shown as a tensor product net (C) and as an arithmetic example (D). A ternary relation is chunked to a binary relation R(a,b/c) in E (with symbol vector omitted for simplicity). A circular convolution calculated from the tensor product in Figure 1B is shown in Figure 1F. The circular convolution is computed by adding along the curved lines and is:

$[0.50\ 0.71\ 0.50] * [-0.5\ \ 0.71\ -0.50] = [-0.25\ -0.25\ 0.00]$

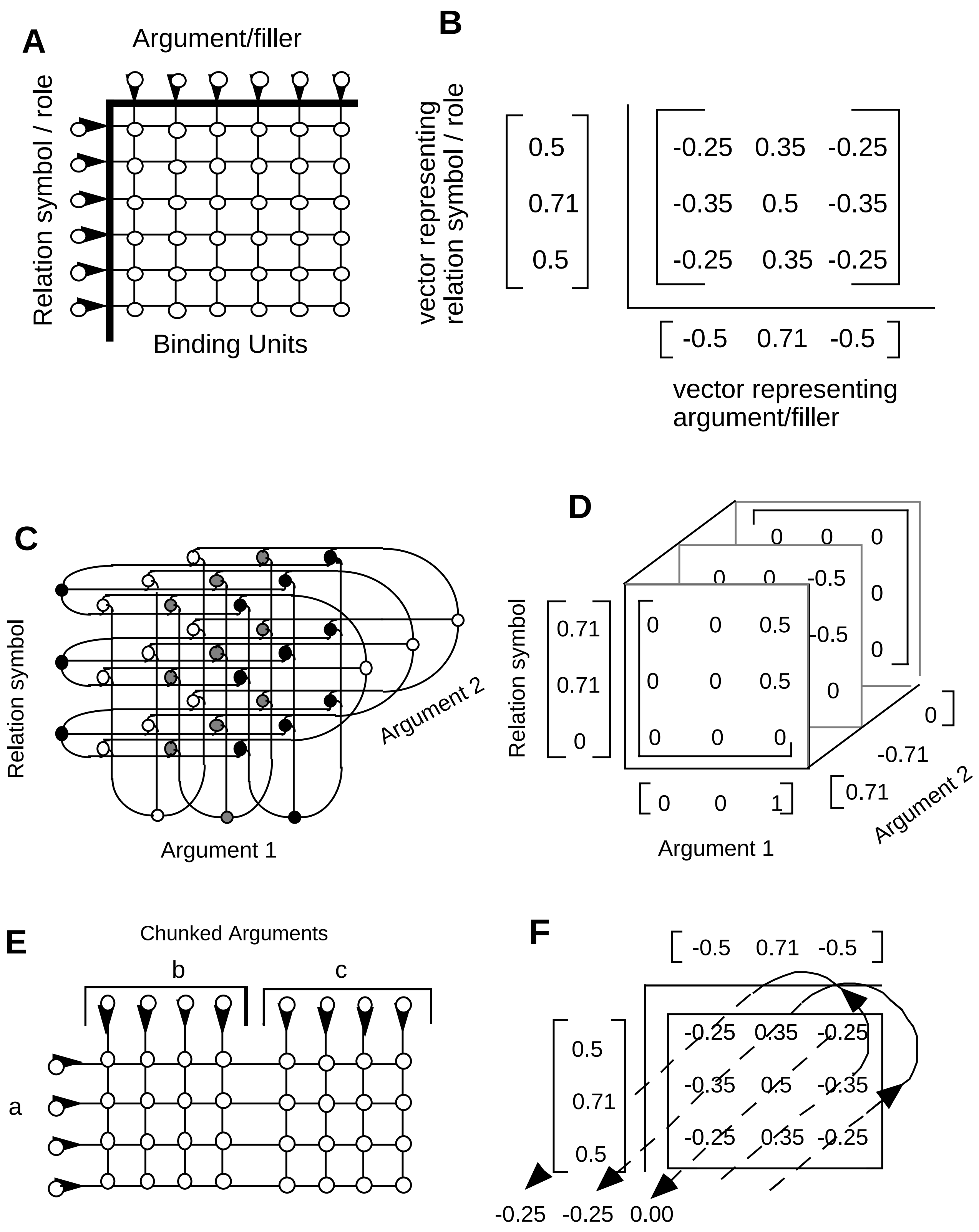
The shadings in Figures A and C are to make the spatial layout clear, and do not represent levels of activation.

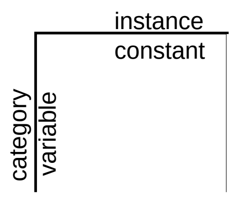
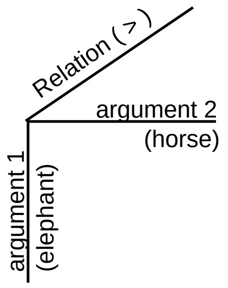
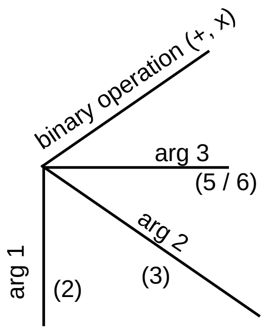
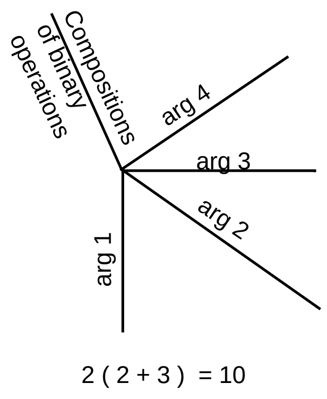
Figure 2. Four levels of relational complexity, with dimensionality and schematic outer product representations.

Figure 3. Synchronous oscillation representations of relational instances love(John,Mary), kiss(John,Mary), and marry(John,Mary).

Figure 4**.** A 3-level binary cascade adder

Figure 5. Decision process for parsing centre-embedded sentence.


A
Argument/filler
Relation symbol / role
Binding Units
B
vector representing relation symbol / role
0.5
0.71
0.5
-0.25 0.35 -0.25
-0.35 0.5 -0.35
-0.25 0.35 -0.25
-0.5 0.71 -0.5
vector representing argument/filler
C
Relation symbol
Argument 1
Argument 2
D
Relation symbol
0.71
0.71
0
0 0 0.5
0 0 0.5
0 0 0
0 0 -0.5
-0.5
0
0 0 0
0
0
0
0 0 1
Argument 1
0.71
-0.71
0
Argument 2
E
Chunked Arguments
b
c
a
F
-0.5 0.71 -0.5
0.5
0.71
0.5
-0.25 0.35 -0.25
-0.35 0.5 -0.35
-0.25 0.35 -0.25
-0.25 -0.25 0.00

Formal Specification	**Dimensionality**	**PDP Implementation**
Unary relations	1	
Binary relations, univariate functions.	2	 Relation (>) argument 2 (horse) argument 1 (elephant)
Ternary relations, binary operations, bivariate functions.	3	 binary operation (+, x) arg 3 (5 / 6) arg 2 (3) arg 1 (2)
Quaternary relations, compositions of binary operations.	4	 Compositions of binary operations arg 4 arg 3 arg 2 arg 1 2 (2 + 3) = 10

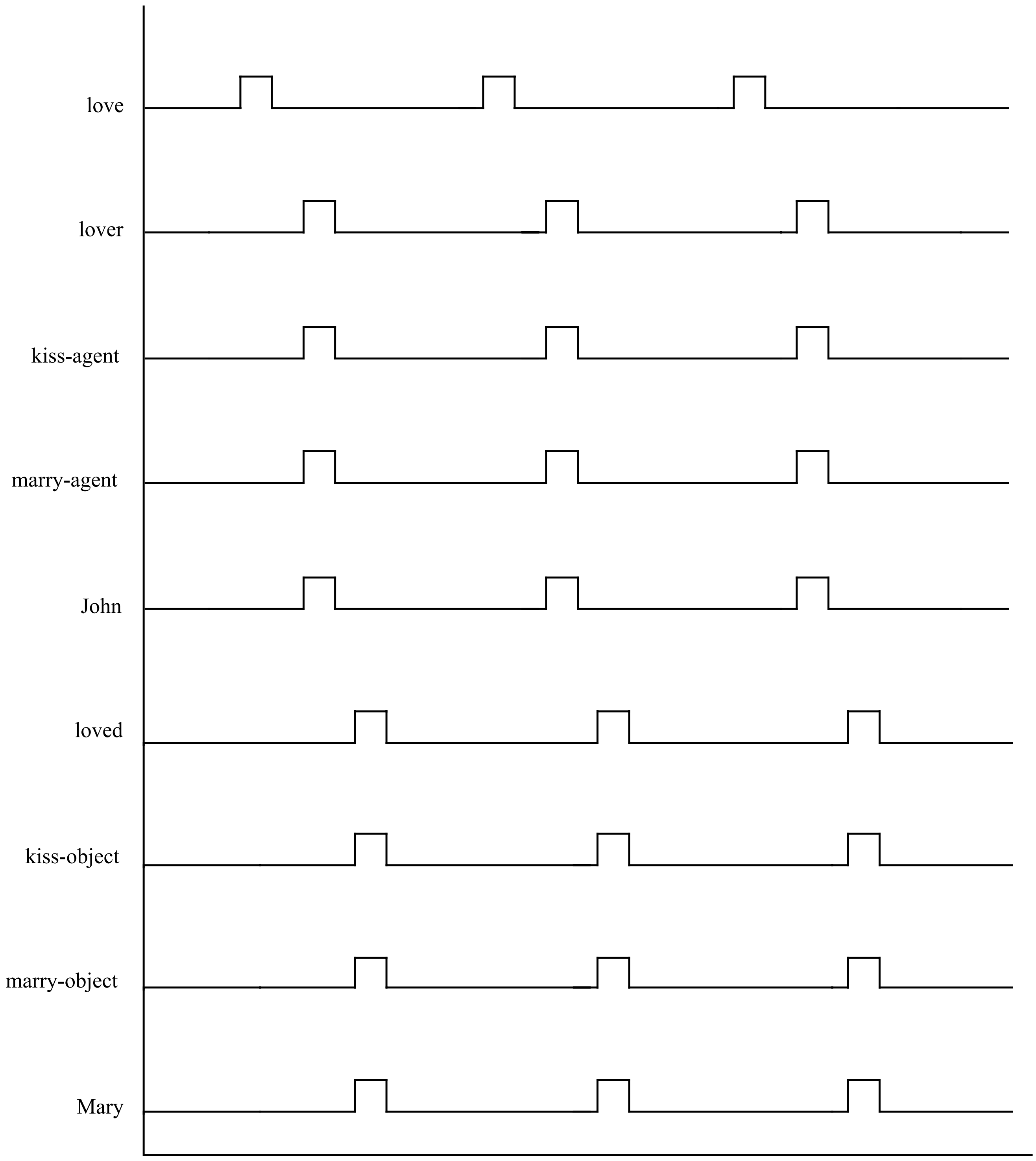


Figure 3. Synchronous oscillation representations of relational instances love(John,Mary), kiss(John,Mary), and marry(John,Mary).

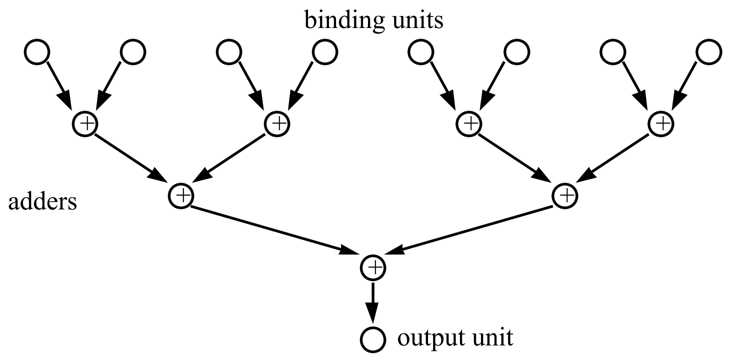


Figure 4. A 3-level binary cascade adder

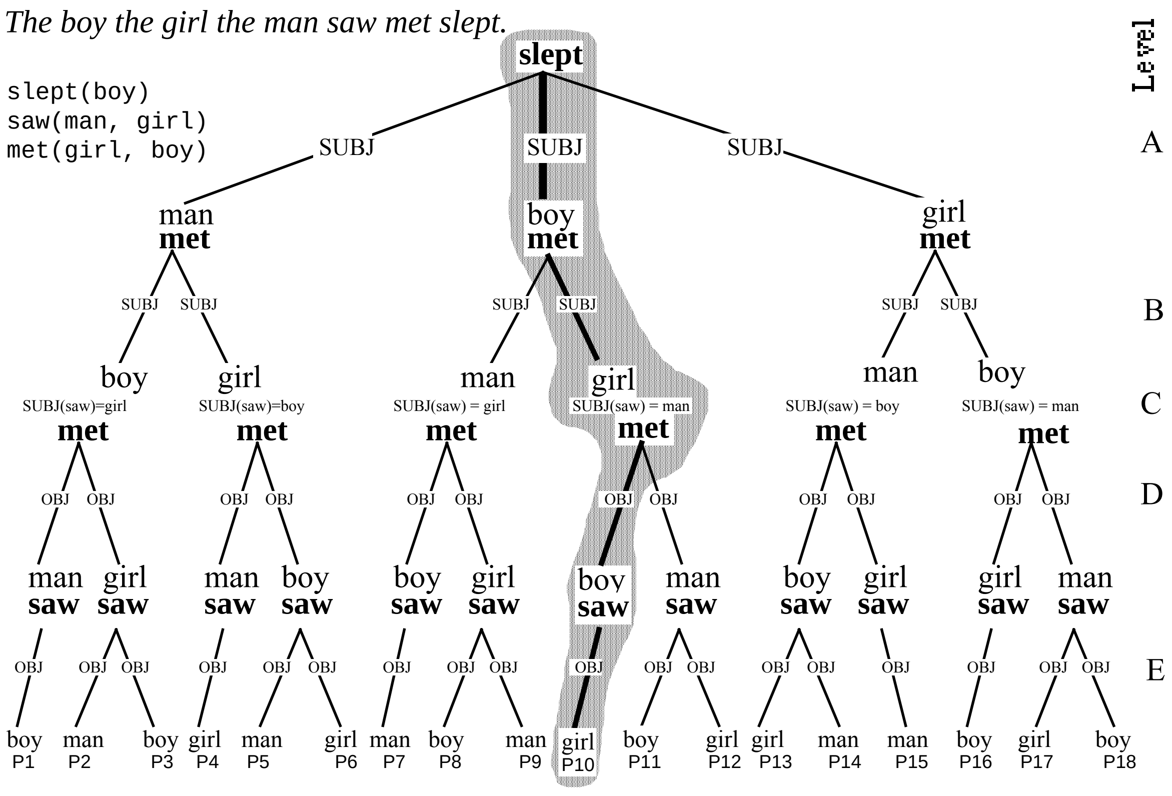


Figure 5. Decision process for parsing centre-embedded sentence.

---

[i] More precisely, “x:dog(x) is bigger than y:cat(y)” has no truth value because of the variables present. However, the corresponding family of propositions referring to particular dogs and cats will be true in some cases (presumably most cases) and false in others, as when x is a Chihuahua and y is a decent-sized cat.

[ii] Note that the structural correspondence principle does not necessarily imply the person in the lover role must always be first in the expression, but it does mean that entities in a given role must always be in the same position relative to other roles. Thus if “John loves Mary” is represented by loves(John,Mary) then “Peter loves Angela” must be represented by l (loves(Peter,Angela), and so on.

[iii] A dimension A is independent of another dimension, B, if, when classifying entities according to their values on dimensions A and B, knowing the value of an entity according to dimension A does not always determine the value of the entity according to dimension B.

[iv]Though their significance could not be determined.

[v] Actually the problems of instance of identification are even greater than this. Not only must each relational instance be bound to a unique context vector (i.e., the context in which the relational instance was memorized), but each vector must be orthogonal (dissimilar) so as to avoid the problem of cross-talk (role vectors bound to fillers from different relational instances). The orthogonality requirement introduces a dilemma: if we choose random dissimilar vectors, then in general relational instances are not distinguishable on the basis of their contents. On the other hand, if we generate identification vectors on the basis of contents, these vectors will no longer be orthogonal since many relational instances share the components. These problems can be overcome by defining each instance by its components and linking them together, but notice that this effectively entails adopting symbol-argument-argument bindings.

[vi] In the special case, where each role is represented by a unique local basis

vector (e.g., [1 0 0], etc), the tensor role-filler method also maps each relation dimension onto a separate tensor subspace. However role-filler methods may entail the psychologically unrealistic assumption that each relational instance can be uniquely identified by a context independent vector.

[vii] This property can be linked to compositionality (Fodor & Pylyshyn, 1988) but it is not appropriate to develop that link in this paper.

[viii] Order notation O(.) identifies the term in *n* with the largest power, so that $100n^2$; $n^2 + 10n$; and $0.001n^2$ are considered to be of the same order (i.e., $O(n^2)$ - quadratic).

[ix] Frye et al. (1995) refer to this as a “setting condition”.