What is modularity good for?

Michael S. C. Thomas (m.thomas@bbk.ac.uk) **Neil Forrester (n.forrester@bbk.ac.uk)**

Fiona M. Richardson (f.richardson@bbk.ac.uk)

Developmental Neurocognition Laboratory,
School of Psychology, Birkbeck College,
University of London, WC1E 7HX UK

Abstract

We compare three types of dual-route associative architectures for learning the English past tense problem. Identical computational resources are used in (1) a *pre-specified modular* architecture, with a rule mechanism and an exception mechanism; (2) an architecture with two mechanisms which demonstrates *emergent specialization* of function for regular and exception verbs; and (3) a system with *redundant* use of its two mechanisms. The *pre-specified modula*r solution was the least efficient for learning the past tense. This was due to difficulties in resolving the competition when its two modules attempted to drive the same output in different ways. The results are discussed in the context of modularity theory.

Introduction

The notion of modularity figures early in the history of cognitive science as a design principle for building complex computational systems. Thus Marr (1982, p.325) argued that 'any large computation should be split up into a collection of small, nearly independent, specialized sub-processes'. Fodor (1983) further developed the principle in the context of cognition, suggesting that modularity is likely to hold sway for low-level sensory and motor systems. For Fodor, modularity represented a probable coalition of processing properties (domain-specificity, informational encapsulation, innate specification, fast operation, hardwired at a neural level, autonomous, not assembled). Modularity saves a low-level system from having to consult all an organism's knowledge in order to do its job, instead acting over a restricted propriety *knowledge base* and potentially employing specialized *processes* (see Fodor, 2000, for the distinction between *epistemological* and *psychological* modularity). From a developmental perspective, a restricted domain of operation also simplifies the learning problem faced by the given sub-system.

Fodor (1983) additionally argued that modularity would not apply to the central cognitive system, where access to background knowledge is available and computations are subject to global constraints of context. Later he argued that the central system might include the majority of cognition, so that modules would have limited explanatory scope (Fodor, 2000). However, others extended the principle of modularity to high-level cognition, under what Fodor refers to as *the massive modularity thesis* (2000). This move was driven both by (1) proposals from evolutionary psychology that humans might inherit domain-specific reasoning systems (e.g., for detecting social cheats, for predicting other people's belief states), and (2) evidence from cognitive neuropsychology of double dissociations between high-level abilities in acquired brain damage. Debates continue about the necessary and sufficient features that define a module (e.g., for Coltheart, 1999, the main feature is domain specificity; Fodor, 2000, prefers encapsulation).

The aim of this article is to consider the computational advantages and disadvantages in opting for modular architectures in systems required to learn different sorts of cognitive problem. While accepting there are innate constraints on the architecture of the cognitive system, our perspective is essentially developmental. By way of illustration, Calabretta et al. (2003) argued that the genotype of behaviorally complex organisms might be more likely to encode modular neural architectures because this avoids possible neural interference. They presented simulations in which a connectionist network was presented with letters on an input retina, and was required either to output *Where* on the retina a letter appeared or *What* letter it was. Table 1 shows different three-layer architectures for systems with common or shared inputs, outputs, and processing resources. Calabretta et al. compared a system with common processing resources (Table 1, panel 5) with a system incorporating modular structure (panel 7). The modular architecture was found to be consistently superior in learning the task. This result arose because information required to compute *Where* is different from that required to compute *What*. There is no advantage in sharing information in a common representational layer. The modular architecture prevents the *What* channel from having to consider irrelevant information from the Where channel and vice versa, thereby aiding the learning process.

Table 1: Architectures with different modular commitments

		OUTPUT			
		Common		Separate	
		PROCESSING RESOURCES			
		Common	Separate	Common	Separate
INPUT	Common	1 	3 	5 	7 
	Separate	2 	4 	6 	8 

In this article, we evaluate the utility of modularity in another domain, English past tense. The domain is of interest because it has a dual structure requiring a child to learn (1) a general regularity, that the past tense of most verbs is formed by adding '-ed' to the stem (e.g., *talk=>talked*), a regularity that is productive for novel verbs (*wug=>wugged*); and learn (2) a restricted set of exceptions to the rule, of various sorts (e.g., *hit=>hit, sing=>sang, go=>went*).

Pinker (1991) proposed that children learn this domain using a modular architecture which comprises a 'computational component containing specific kinds of rules and representations' and an 'associative memory system with certain properties of connectionist models' (1999, p.531), which learn the past tense rule and the exceptions, respectively. The rule-component operates as the default, while for exceptions, the memory component blocks the rule mechanism and delivers the exception form. Key empirical data indicate that children pass through an extended phase of 'over-regularization' where the rule is mistakenly applied to exception verbs (e.g., *think=>thinked*), suggestive of interference between two mechanisms. A debate continues on the status of this theory (see Thomas & Karmiloff-Smith, 2003, for a review).

Our interest here is not to enter into this debate per se, but to use computational simulations to explore whether (and how) modular solutions offer an advantage for acquiring the past tense domain. We will begin with two assumptions. Assumption 1: the problem can be defined as one of learning the mapping between phonological representations of the verb stem and past tense form (this assumption could be wrong; see Thomas & Karmiloff-Smith, 2003). Assumption 2: the developmental system has two learning mechanisms available to it, one with computational properties better suited to learning regular mappings and one able to learn potentially arbitrary exceptions to the rule. Our architecture corresponds to Table 1, panel 3. Given our two mechanisms, it is important to realize that there are at least three ways to combine them that make different modular commitments. Diverse computational components do not themselves define a modular architecture.

To determine the architecture, one must answer three questions. First, do input patterns get separately channeled (by some gatekeeper knowing about regulars and exceptions) to the different mechanisms? Second, do the mechanisms compete to drive the output, or can they collaborate in producing a response? Third, are the two mechanisms given equal opportunity to learn the problem, or does the improving performance of one mitigate the need for the other to improve its accuracy? We refer to these three dimensions, illustrated in Figure 1, as Input competition, Output competition, and Update competition, respectively (Thomas & Richardson, 2006). Depending on these three choices, the same processing resources can be used to create a *pre-specified modular* system (inputs channeled, components compete to drive output); a system demonstrating *emergent specialization of function* of its components (components learn the parts of the task for which their computational properties are fitted via update competition alone); or a *redundant* system (both components attempt to learn all the task and compete to drive output). Thomas and Richardson (2006) demonstrated that both modular and emergent solutions exhibited double dissociations between regular and exception verbs in the endstate, although dissociations were stronger in the modular case; the redundant system only showed single dissociations.

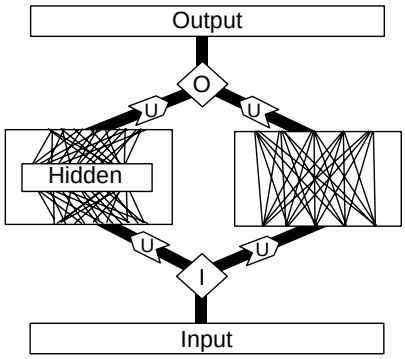

Figure 1: Use of (I)nput, (U)pdate and (O)utput competition to create a modular, emergent, or redundant system using the same two components

Decisions about modularity are not, therefore, simply about combining components with different domain-specific computational properties – in this example, the same components and properties deliver different modular solutions. Restricted information flow is as important, and indeed may deliver pre-specified modularity on its own if the components share common processing properties.

So our research question becomes, of the three ways of using the same resources, is the modular one the best? In our investigations, both our learning mechanisms were associative; respectively, a two-layer and a three-layer connectionist network. The two-layer network is better for learning regular mappings (faster, better generalization), while the three-layer network is better able to learn potentially arbitrary associations.

Pinker's dual mechanism model

We should make clear our simulations neither sought to implement nor to test Pinker's (1991) dual mechanism model of past tense acquisition. Explicitly, although we explored a modular architecture with a rule-learning component and an exception learning component, the rule-learning component we used is not that intended in Pinker's theory, since the latter mechanism remains insufficiently specified to allow implementation. The aim of the current simulations was to begin to address the issues that Pinker's theory raises with regard to modularity by using a readily available associative network optimized to learn regular mappings as a proxy for a proper rule-learning mechanism. A few comments about Pinker's theory will make this point clearer.

The blocking principle: In Pinker's theory, the exception mechanism blocks the operation of the rule when an exception past tense form is retrieved from memory for a given verb stem (see Marcus et al., 1992, p.8-18, for details). Retrieval failures explain the occasional interference errors between the mechanisms. These 'over-regularization errors' (e.g., *thinked*) occur predominantly (but not exclusively) in childhood. Marcus et al. (1992) are clear that the idea of blocking is not based on developmental evidence but derived from adult linguistic theory and simply attributed to the child (p. 16). It has not been implemented.

The rule-learning mechanism: Marcus (2001) points out that the intended rule is a symbolic operation specified over the variable *verb stem*. The acquisition of this rule has not been clearly explicated, but appears to invoke both inductive and deductive inferential processes. For example, Marcus et al. (1992) list possible cues that the children might look for to recover the rule (e.g., the heterogeneity of stems that are heard to be regularly inflected, p.134). The inflectional system seeks a single rule, or possibly multiple rules (p.133). Pinker (1999, p.194) describes the child's discovery of the rule as a sort of 'epiphany'. Currently available rule-induction algorithms do not seem appropriate to capture the intended process. The rule-learning algorithm must explain the observed gradual improvement in regular past tense formation and also the strong generalization of the '+ed' regularity to novel stems that bear little similarity to those that children know (e.g., *ploamph=>ploamphed*; Pinker, 1991; see Marcus, 2001, for discussion). Operation of the rule-learning device awaits further specification.

The Revised Dual Mechanism (DM) model: In 1999, Pinker revised his model to weaken its modular commitments. In the new model, the rule mechanism attempts to derive the past tense rule, while the lexical memory attempts to learn (potentially) all the past tenses. These might include regulars that are high frequency or sound similar to distracting exceptions (e.g., *blinked=>blinked*, *think=>thought*). Here one mechanism has a restricted remit (regulars) while the other has a full remit (all verbs), creating a partially redundant architecture. Some have argued that this redundancy accounts for residual past tense acquisition in children with Specific Language Impairment (see Thomas, 2005, for discussion).

Simulations

The simulation section will proceed as follows. We first briefly introduce details of the architectures, training set, and parameters. We then compare developmental trajectories for our *modular*, *emergent*, and *redundant* systems on the past tense problem, considering both performance on the training set, interference errors, and generalization to novel verb stems. Where the exception mechanism was required to learn the full training set, its level of resources turned out to be crucial, and so results are presented for exception mechanisms with low and high resources. Among the high resource conditions, we consider a partially redundant architecture similar to the *Revised DM* model. Lastly, we will find that the three varieties of modular system (low resource, high resource, and Revised DM) present difficulties in coordinating the output of their two mechanisms, and so we consider adjustments to these models to optimize their performance.

Simulation details

Architecture: The network had 90 input units and 100 output units. The 'rule' mechanism comprised a 2-layer network directly connecting input and output units. The 'exception' mechanism comprised a 3-layer network, with a layer of hidden units interceding between the input and output layers. Twenty hidden units were used in the low resource condition and 100 in the high resource condition.

Training set: The training set was based on the simplified rendition of the past tense problem used by Plunkett and Marchman (1991). Verb stems were triphonemic consonant-vowel strings encoded using binary phonetic features. Thirty units encoded each phoneme and the outputs layer included an additional 10-unit inflection morpheme. There were 410 regular verbs, 20 no-change exceptions, 68 vowel-change exceptions, and 10 arbitrary exceptions. Hereafter, the exceptions are labeled EP1, EP2, and EP3f, respectively. Training items were split into high and low frequency groups. To ensure the acquisition of arbitrary exceptions, these were given a higher token frequency than all other patterns, marked by the 'f'.

Generalization set: Novel stems could either share two phonemes with existing verbs (rhymes) or only one phoneme (non-rhymes). There were 410 regular rhymes, 10 EP1 rhymes, 76 EP2 rhymes, 10 EP3f rhymes, and 56 non-rhymes. We report extension of the rule to regular rhymes, referred to as *rule(sim)*; extension of the rule to non-rhymes bearing low similarity to any stem in the training set, referred to as *rule(nosim)*; extension of the rule to EP2 rhymes (e.g., *ling=>linged*); and irregularization of EP2 rhymes (e.g., *ling=>lang*).

Competition mechanisms: Input competition was implemented by training the 2-layer network and the 3-layer network separately on regulars and exceptions respectively. It therefore assumes a type of input gatekeeper (see Fodor, 2000, p.71-78, for discussion). For Update competition, each mechanism was backpropagated with error signals from the output generated by both mechanisms combined; for no Update competition, each mechanism received error signals from its own output response alone. To capture Output competition, the output of each mechanism was assigned a 'confidence' value reflecting how binary the vector was (since all targets were binary feature sets). Formally, the output vector was thresholded at 0.5 (if $x<0.5$, $x=0$; if $x>0.5$, $x=1$) and the Euclidean distance was derived between actual and thresholded versions. The mechanism with the highest confidence was assigned the winner and drove the final output. Without Output competition, the output of each mechanism was summed to create the net input to the output layer.

Parameters: Models were trained using the backpropagation algorithm with a cross entropy error measure, learning rate of 0.1, momentum of 0, for 500 epochs (random order without replacement). The full training set was used rather than an incrementally increasing set, so these simulations do not aim to capture an early high performance on a restricted set of regular and exception verbs. Performance was measured at 1, 2, 5, 10, 25, 50, 100, 200, and 500 epochs of training. Six replications of each network were run using different random seeds. Error bars are omitted from figures for clarity but all reported differences are reliable.

Results

We begin with the developmental trajectories generated by each system. Figure 2 compares modular, emergent, and redundant systems when the exception mechanism has low resources. The modular condition generated fast learning of regulars and high generalization of the rule, even to novel stems bearing low similarity to anything in the training set (*sim*: 97%, *nosim*: 65%). Pinker (1991, p.532) implies that *rule(sim)* and *rule(nosim)* generalization should be at the same level, suggesting our proxy rule learning mechanism is not sufficiently powerful for the DM account. However, the modular system could not learn the exceptions; the rule mechanism was always more confident of its answer than the exception mechanism because it was learning a more function. The redundant system learnt more evenly but did not reach ceiling on either regulars or exceptions because the rule mechanism didn't have the power and the exception mechanism didn't have the resources to learn the whole problem. The emergent system reached ceiling on regulars and exceptions, but with generalization at 84% (*sim*) and 31% (*nosim*).

When the exception mechanism was given higher resources, the modular system still failed on exceptions for the same reason, although there was no some presence of the exceptions in the output, especially for EP3f. Both emergent and redundant systems reached ceiling and showed comparable generalization (*sim*: 87 vs 85%, *nosim*: 32 vs 28%). The modular system retained its much higher generalization (*sim*: 97%, *nosim*: 61%). The Revised DM condition, with an exception mechanism trained on both regulars and exceptions performed little different to the modular.

Figure 2 bottom panel depicts interference errors (over-regularization of exceptions) for each exception type across training, for all systems. All systems exhibited these errors, and all showed the comparatively reduced vulnerability of the higher frequency EP3f patterns. For modular and Revised DM systems, the errors never went away. Interference errors per se, therefore, are not diagnostic of architecture. Of course, their exact timing and proportions may be in a detailed comparison to empirical data, although that is not the aim of the current simulations.

Let's try and fix the modular systems. Exception mappings are more complicated, so the rule mechanism is always likely to be more confident of its regular response than the exception mechanism is of its (mostly) unsystematic transformations. One way to fix the problem is to bias the output of the exception mechanism, amplifying its confidence level. Figures 3-5 show the change in developmental trajectories that different levels of biasing produced, for the low resource modular, high resource modular, and revised DM respectively. In each case, results are split into training and generalization. The bias factor simply multiplied the confidence value of the exception mechanism by a fixed value. We plot trajectories for biases of x1 (original), x2, x5, x10, x50, x100, x200, x250, and x1000.

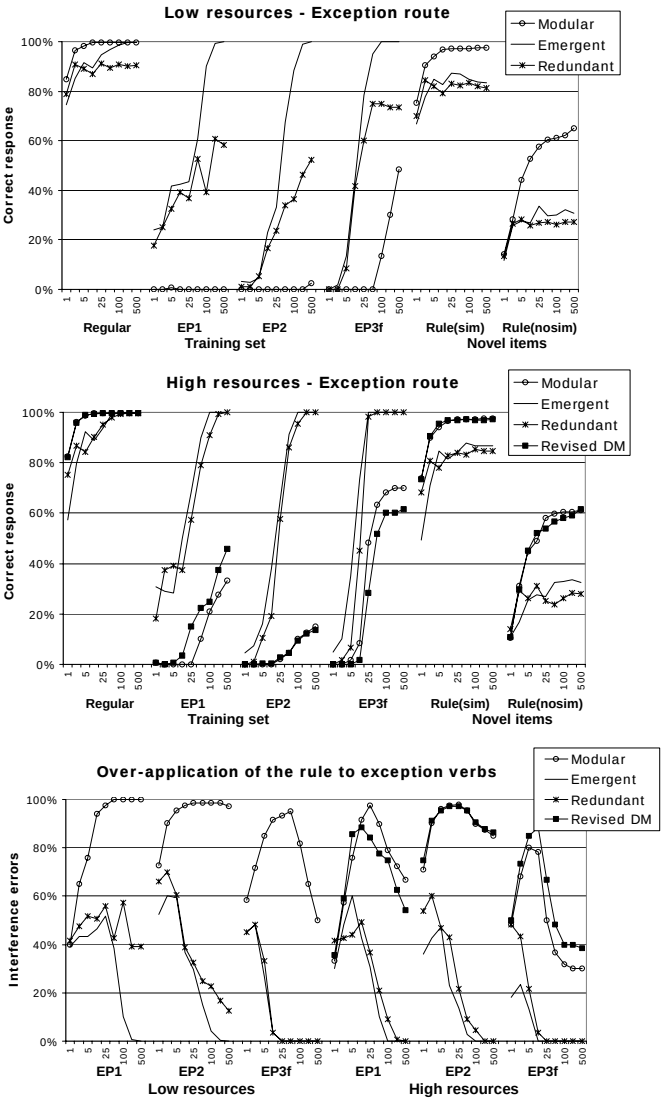

Figure 2: Developmental trajectories and interference errors for the different architectures

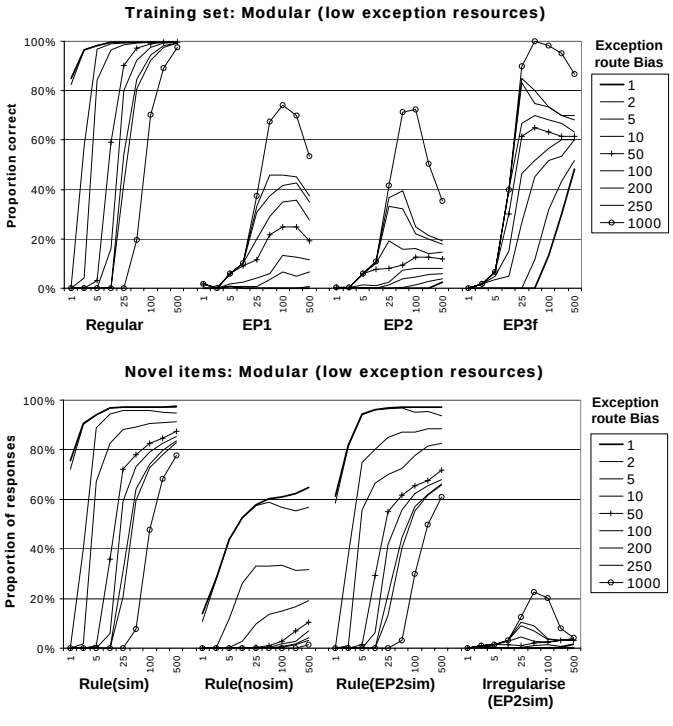

Figure 3: trajectories for the modular system with biasing to increase the role of the exception mechanism

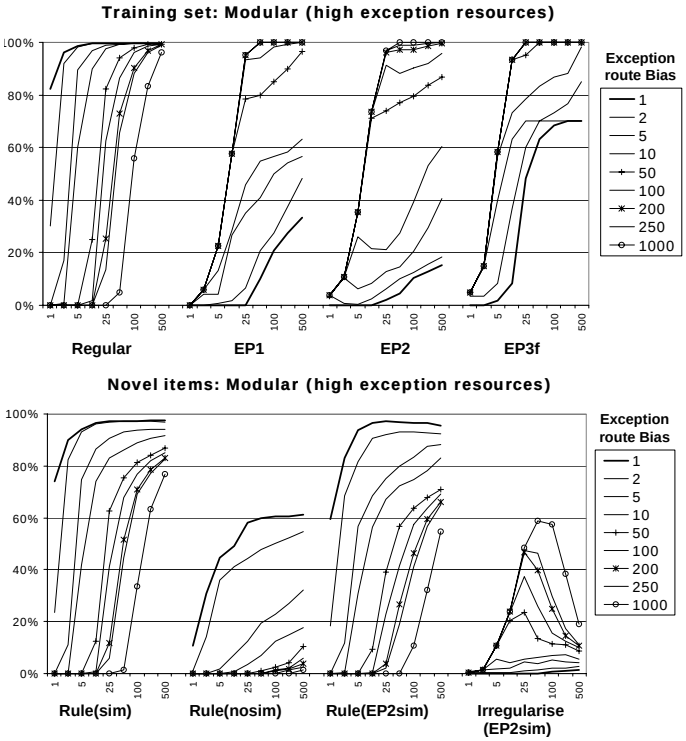

Figure 4: biased trajectories for the modular system

This way of fixing the modular systems may seem post-hoc, but one can imagine how an optimal biasing value for Output competition might be derived during training. The bias starts at 1 and is increased (by some small amount) each time the exception mechanism has the correct output but fails to block operation of the rule mechanism.

None of the bias values considered were sufficient to allow exceptions to be learned in the low resource modular system (Figure 3). Notably, as exception bias values were increased, regular learning slowed, rule generalization decreased, and irregularization of novel stems (e.g., *ling=>lang*) increased. *Nosim* generalization, the key domain of the rule mechanism, collapsed as soon as biasing exceeded x2.

In the high resource condition, the modular system reached ceiling performance by the end of training when the exception bias was x200 (marked by asterisks in Figure 4). At this bias level, generalization for *rule(sim)* was 83%. By comparison, for the emergent system it was 87% and for redundant 85%. For *nosim*, the modular was 4%, the emergent was 32% and the redundant was 28%. Acquisition of regulars was much slower for the biased high resource modular system compared to emergent and redundant solutions, but its acquisition of exceptions was faster.

Finally, the partially redundant Revised DM condition revealed a similar pattern to the high resource modular on the training set. However, since the exception mechanism was now required to learn the whole training set, its confidence needed greater amplification. Performance was just under ceiling with a bias of x1000.

The main difference between Revised DM and high resource modular was that the former did not experience the marked slowing in regular verb acquisition, or reduction in generalization. Final *sim* generalization of the rule was 90%, slightly higher than emergent and (fully) redundant. This marginal increase in generalization was the sole benefit of the rule-dedicated mechanism. (*Nosim* was at a comparable 28%). The generalization advantage stemmed from the fact that while the influence of the rule mechanism is initially reduced early in training (as for the high resource modular), some of this function was taken up early in training by the exception mechanism, which is itself able to generalize the rule. Figure 5 demonstrates the relative influence of the two mechanisms in driving regulars and rule generalization.

Discussion

Modular solutions to learning the past tense were problematic because the component mechanisms generated different outputs for the same input, and the competition between the mechanisms then had to be resolved. While redundant architectures also required the settling of this competition, the mechanisms were more often than not offering similar outputs. What the modular system gained by including a dedicated rule mechanism, it then lost in mediating the competition between its two mechanisms. For the exception mechanism to speak loud enough to block the rule mechanism, it had to eat into the generalization offered by the rule mechanism. Both emergent and redundant solutions were more successful developmental solutions. The emergent was most efficient in terms of resources

because its sole reliance on Update competition encouraged cooperation between its two components. Interference errors in children (e.g., *thinked*) have been seen as diagnostic of a modular solution and faulty blocking. However, these errors appeared in emergent and redundant architectures as well.

Our results are consistent with a simulation result reported by Calabretta et al. (2000). When these authors trained a robot to learn a sensorimotor task, duplication of partially adapted modules greatly facilitated evolution of functional specialization. But there was no evidence that functionally specialized modular systems had inherently better performance or were more trainable than non-specialized modular systems. In the language domain, our results are reminiscent of those of Hahn and Nakisa (2000) in a model learning the default German plural. Addition of an explicit rule did not aid generalization (although in this case, the rule mechanism was not an integrated developmental element of model). Our findings do not serve to undermine Pinker's (1991) dual mechanism model of past tense formation because our simulations were not an implementation of this theory. We used an associative rule-learning mechanism rather than a symbolic rule acting on the stem as a variable (whose implementation is as yet unclear). Our findings suggest the nature of 'blocking' will be key for the operation of an implemented version when it arrives.

Conclusion

What is modularity good for? When processing components drive separate outputs and the information required by each output is independent, modular developmental solutions may be optimal (Calabretta et al., 2003). When processing components receive information from a common input and have to drive a common output, a pre-specified modular architecture may be inefficient, since it is necessary to resolve a competition for which module will drive output. Either an emergent or redundant solution using the same resources may be superior. For the problem domain considered, cooperation is more efficient than competition.

Acknowledgments: This research was supported by UK MRC CE Grant G0300188 to Michael Thomas

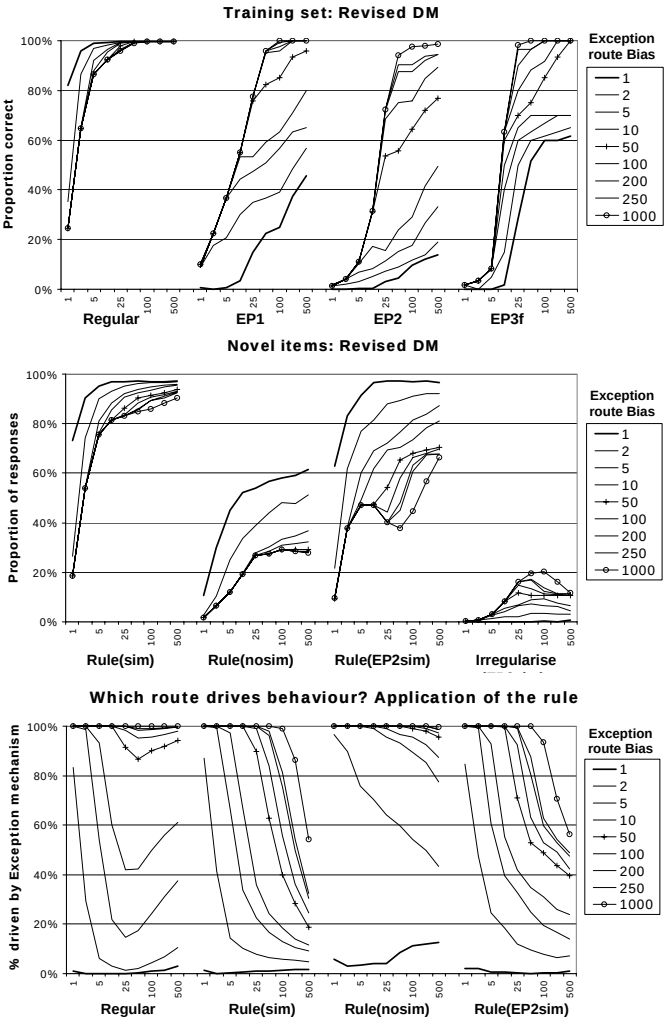

Figure 5: biased trajectories for the Revised DM condition

References

Calabretta, R., Di Ferdinando, A., Wagner, G. P., & Parisi, D. (2003). What does it take to evolve behaviorally complex organisms? *Biosystems, 69*, 245-262.

Calabretta, R., Nolfi, S., Parisi, D., & Wagner, G. P. (2000). Duplication of modules facilitates the evolution of functional specialization. *Artificial Life, 6(1)*, 69-84.

Coltheart, M. (1999). Modularity and cognition. *Trends in Cognitive Sciences, 3(3)*, 115-120.

Fodor, J. A. (1983). *The modularity of mind*. Cambridge,

Fodor, J. A. (2000). *The mind doesn't work that way: The scope and limits of computational psychology*. MIT Press.

Hahn, U., & Nakisa, R.C. (2000). German Inflection: Single or Dual Route? *Cognitive Psychology, 41*, 313-360.

Marcus, G. F. (2001). *The algebraic mind: Integrating connectionism and cognitive science*. MIT Press.

Marcus, G., Pinker, S., Ullman, M., Hollander, J., Rosen, T. & Xu, F. (1992). Overregularization in language acquisition. *Monographs of the Society for Research in Child Development, 57* (Serial No. 228).

Marr, D. (1982). *Vision*. W. H. Freeman.

Pinker, S. (1991). Rules of language, *Science, 253*, 530-535.

Pinker, S. (1994). *The Language Instinct*. Penguin books.

Pinker, S. (1999). *Words and rules*. London: Weidenfeld & Nicolson

Thomas, M. S. C. (2005). Characterizing compensation. *Cortex, 41(3)*, 434-442.

Thomas, M. S. C. & Karmiloff-Smith, A. (2003). Modeling language acquisition in atypical phenotypes. *Psychological Review, 110(4)*, 647-682.

Thomas, M. S. C., & Richardson, F. (2005). Atypical representational change: Conditions for the emergence of atypical modularity. In M. Johnson & Y. Munakata (Eds.), *Attention and Performance XXI*. Oxford: Oxford University Press.