

EPIC: An Embodied Cognitive Architecture for Modeling Human Cognition and Performance

David Kieras & David Meyer

University of Michigan

March 22, 2003

Sponsor: ONR

Introduction

A Cognitive Architecture:

The EPIC Architecture

Diagram of the Current EPIC Architecture

Example Structural Detail - Visual System

Goals of EPIC Project

Distinctive Features of EPIC Work

Importance of Perceptual-Motor Constraints

Some Perceptual-Motor Constraints in EPIC

Central Mechanisms

Sample Rules - 1

Sample Rules - 2

A Cognitive Architecture:

A fixed set of components and mechanisms that represent basic human abilities and limitations - task independent.

Change only if pervasive and hoped to be permanent.

The architecture is “programmed” with a strategy to perform specific tasks.

Architecture provides constraints and possibilities for the form and content of the strategy.

Architecture + a specific strategy = a model of a specific task.

Should predict performance.

The EPIC Architecture

Basic assumptions

- Production-rule cognitive processor.
- Parallel perceptual and motor processors.

Fixed architectural properties

- Components, pathways, and most time parameters

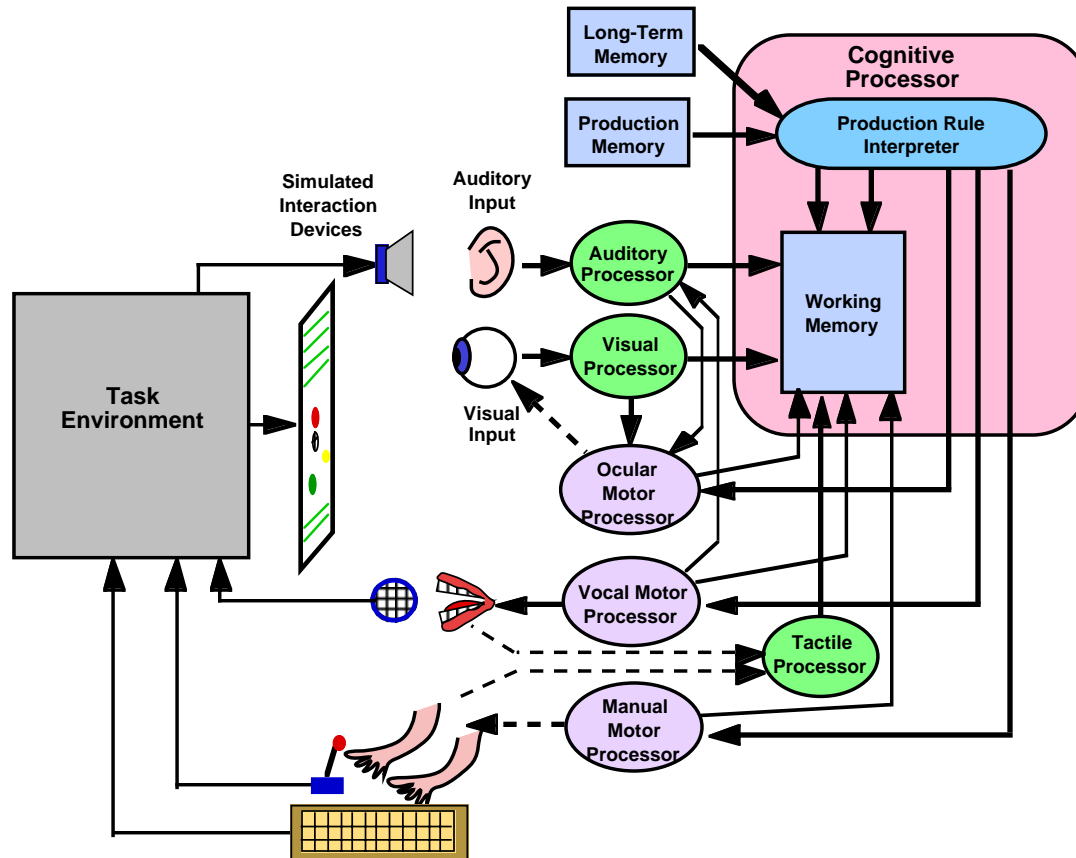
Task-dependent properties

- Cognitive processor production rules.
- Perceptual recoding.
- Response requirements and styles.

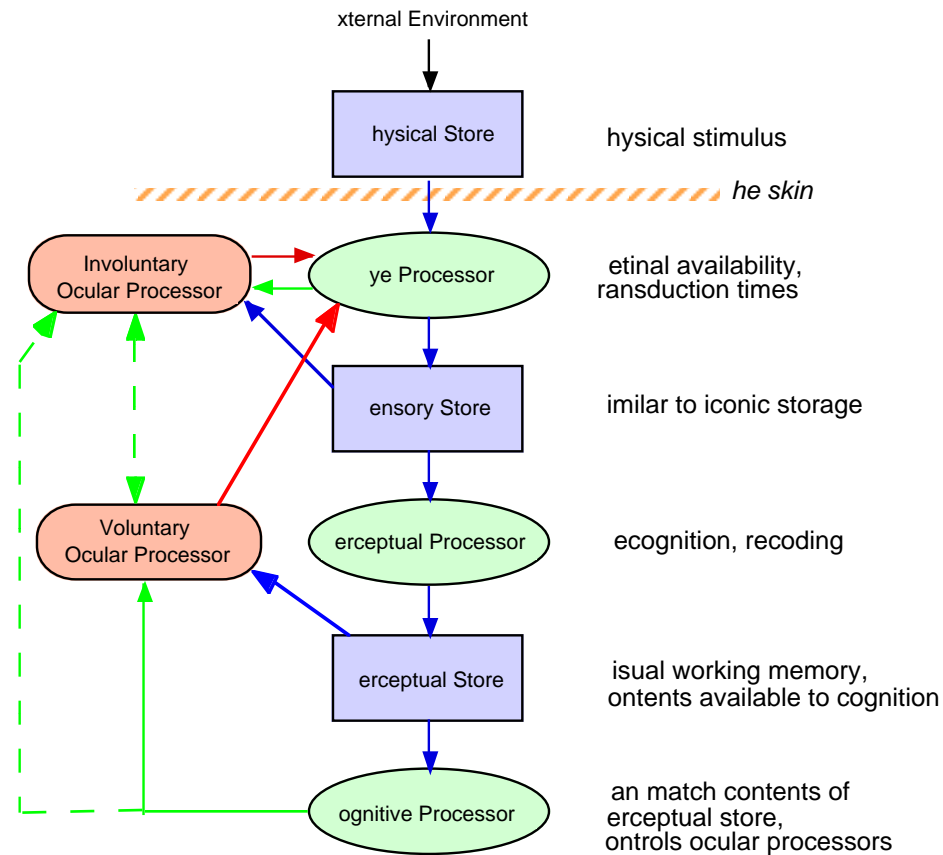
Currently, a performance modeling system.

- Theory of human performance not exactly finished - plenty of work still to be done!
- But learning mechanisms being planned - summarize issues.

Diagram of the Current EPIC Architecture



Example Structural Detail - Visual System



Goals of EPIC Project

Develop a predictive and explanatory theory of human cognition and performance.

Codify scientific knowledge.

Elucidate executive processes.

Explain multitask performance.

Make it accurate and practical enough to use for simulated humans in system design methodology.

Simulate the human-machine system; iterate machine design to achieve required system performance.

Compare to parallel development of a GOMS modeling system for HCI design.

Distinctive Features of EPIC Work

Emphasis on executive processes that coordinate multitask performance.

Multitask performance stresses the architecture.
An important but underdeveloped area for theory.

Take advantage of underexploited but powerful constraints:

Perceptual-motor abilities and limitations.
Detailed and exact quantitative fits to human data.

“Zero-based” theoretical budget:

Question traditional assumptions.
Do not add a mechanism until it is needed to account for data.
Avoid egregious assumptions of cognitive limitations.
Prefer strategy limitations over architectural ones.

Focus on major phenomena and mechanisms that are important determinants of performance, rather than minor “interesting” ones.

Compare multiple strategies for doing a task.

Isolate strategy effects from architectural properties.

Importance of Perceptual-Motor Constraints

In many tasks, performance is primarily limited by peripheral perceptual-motor activities rather than central cognitive limitations.

Account for many key issues in a variety of tasks.
Analogous to traditional bottlenecks in computing.

Ignoring can result in absurd models.

Can ignore only in very heavily cognitive tasks.

Some Perceptual-Motor Constraints in EPIC

Visual resolution depends on eye position, specifics of visual properties.

Different eye movement types and timing.

Different hand movement types and timing.

Cross-constraints for visually-aimed movements.

Hands bottlenecked through single processor, unless two-hand movement style has been learned.

Auditory and speech input recognition timing.

Speech output timing.

Verbal working memory uses auditory and vocal processors, thereby limited by decay and production rate properties.

Visual working memory appears to have large short-term capacity, small but reliable long-term capacity under cognitive control.

Central Mechanisms

Perceptual-motor stores and processors operate in full parallel with cognitive processor.

Cognitive processor uses Parsimonious Production System

Very simple syntax and semantics, rete match implementation.

Memory items are simply lists of symbols.

Match & fire in cycles, 50 ms period.

Production rules can fire in parallel.

Any number can fire during a cycle.

All rules whose conditions match will fire.

All instantiations of a rule's condition will fire.

No implicit flow-of-control mechanisms.

No "hard-wired" goal stack, data refractoriness, etc.

Flow of control must be explicit in rules.

Sample Rules - 1

```
(Top-see-fixation-point
  If
  (
    (Goal Do Visual_search)
    (Step WaitFor Fixation-present)
    (Visual ?fixation-point Shape Cross_Hairs)
    (Visual ?fixation-point Color Red)
  )
  Then
  (
    (Add (Tag ?fixation-point fixation-point))
    (Delete (Step WaitFor Fixation-present))
    (Add (Step WaitFor probe-present))
  )
)
```

Sample Rules - 2

```
(Top-make-response
  If
  (
    (Goal Do Visual_search)
    (Step Make Response)
    (Tag ?target target)
    (Tag ?cursor cursor)
    (Motor Manual Modality Free)
  )
  Then
  (
    (Send_to_motor Manual Perform Ply ?cursor ?target Right)
    (Delete (Step Make Response))
    (Add (Step Make Response2))
  )
)
```

Programming EPIC

Flow of control must be explicit in the rules, not implicit in the architecture.

Basic Programming Approach

Hierarchical-Sequential

Hierarchical-Overlapping

Multiple Subgoal Threads

Multiple Rule Steps

Multiple Step Threads

Demons

**Flow of control must be explicit in the rules,
not implicit in the architecture.**

Sequential processing: Each rule disables itself and enables the next rule in sequence.

Goal-directed processing: Each rule governed by a goal condition.

A variety of flow controls are possible and useful.

Basic Programming Approach

Procedures represented as GOMS-like methods.

A set of rules controlled by the same goal condition.
Sequence controlled by step conditions.

Demons: Free-standing rules.

Can fire at any time.
Easily provide interruptability.

Multi-threaded processing:

Multiple goals, steps can be present.
Multiple rules fire independently.
Chains of rules can be spawned, terminated at will.

Hierarchical-Sequential

```
(Top1
  If ((Goal X)(Step x A))
  Then ((Delete (Step x A))(Add (Step x B)))
)
(Top2
  If ((Goal X)(Step x B))
  Then ((Add (Goal Y))(Delete (Step x B))(Add (Step x C)))
)
(Top3
  If ((Goal X)(Step x C)(Not(Goal Y))
  Then ((Delete (Step x C))(Add (Step x D)))
)
.....
(Sub1
  If ((Goal Y) Not(Step y ???))
  Then ((Add (Step y A)))
)
(Sub2
  If ((Goal Y) (Step y A))
  Then ((Delete (Step y C))(Add (Step y B)))
)
(Sub3
  If ((Goal Y) (Step y B))
  Then ((Delete (Step y C))(Delete (Goal y)))
)
```

Hierarchical-Overlapping

(Top1
 If ((Goal X)(Step x A))
 Then ((Delete (Step x A))(Add (Step x B)))
)

(Top2
 If ((Goal X)(Step x B))
 Then ((Add (Goal Y))(Delete (Step x B))(Add (Step x C)))
)

(Top3
 If ((Goal X)(Step x C))
 Then ((Delete (Step x C))(Add (Step x D)))
)

.....

(Top8
 If ((Goal X)(Step x H)(Not(Goal Y))
 Then ((Delete (Step x H))(Add (Step x I)))
)

Multiple Subgoal Threads

```
(Top1
  If ((Goal X) (Step x A))
  Then ((Add (Goal Y))(Add (Goal Z)
        (Delete (Step x A))(Add (Step x B)))
  )
```

```
(Top2
  If ((Goal X)(Step x B)
      (Not(Goal Y)(Not(Goal Z)))
  Then ((Delete (Step x B))(Add (Step x C)))
  )
```

Multiple Rule Steps

```
(Top1
  If ((Goal X)(Step x A))
  Then ((Delete (Step x A))(Add (Step x B)))
)
```

```
(Top2B1
  If ((Goal X)(Step x B) /* some condition */)
  Then /* some action */
)
```

```
(Top2B2
  If ((Goal X)(Step x B) /* some condition */)
  Then /* some action */
)
```

```
(Top2B3
  If ((Goal X)(Step x B))
  Then ((Delete (Step x B))(Add (Step x C2)))
)
```

Multiple Step Threads

```
(Top1
  If ((Goal X)(Step x A))
  Then ((Delete (Step x A))(Add (Step x B1))(Add (Step x B2)))
)
```

```
(Top2B1
  If ((Goal X)(Step x B1))
  Then ((Delete (Step x B1))(Add (Step x C1)))
)
```

```
(Top2B2A
  If ((Goal X)(Step x B2))
  Then ((Delete (Step x B2))(Add (Step x C2)))
)
```

Demons

```
(Demon1
  If ((Visual ?obj Color Red))
  Then (/some action */)
)
```

```
(Demon2
  If ((Goal Process Blip)
      (Tag ?blip Current_blip)(Visual ?blip Status Disappearing))
  Then ((Delete (Goal Process Blip))(Add (Goal Abort Processing)))
)
```

Executive Processes

Explicit Executive Processes

Executive Functions

Executive Regimes

Explicit Executive Processes

“Executive process” - in experimental psychology: process that controls overall behavior - the “homunculus”.

Simply additional production rules - no special-purpose mechanisms required.

Executive process rules manipulate goal conditions and other information to control other sets of production rules.

E.g Dual Task

Rules for Task 1

Rules for Task 2

Executive rules

- Control task 1 & 2 rules as required to meet overall requirements for task coordination.

Executive skills must be learned just like specific task skills.

Executive Functions

Similar to an Operating System:

Enforce sequence, priorities of tasks based on current situation.

Maximize throughput by enabling parallelism where possible.

Allow separately-acquired or separately-usable tasks to be executed concurrently or not, depending on situation.

Mechanisms:

Suspend/resume a task by removing/replacing its governing Goal.

Allocate resources to tasks by inserting/removing permission items.

Executive Regimes

General Executive.

Coordinates any set of independent tasks.

First-come, first-served sequential execution.

Time-splitting.

Concurrent execution, resource management.

Specialized Executive.

Coordinates two specific tasks.

Preallocation of resources.

Task-specific optimizations.

Specialized single multitask program - "flat" model.

No distinct executive rules; task rules do coordination.

Elimination of executive overhead.

Task-specific optimizations.

Strong effects of different executive regimes.

E.g. whether processing can be overlapped between tasks.

Requirements for form/organization of task rules.

Possible learning implication: Improve performance by changing executive regimes.

Learning

Learning Multitask Skill

Simplest Multiple Task: Overlapping Choice Reaction Tasks

Single-task Learning

Color-finger mapping Mean RTs

Compatible spatial-finger mapping Mean RTs

Incompatible spatial-finger mapping Mean RTs

Incompatible spatial-finger mapping IQR RTs

Dual-task Learning

Compare a Good and Poor Time-sharer

Good Time-sharer Mean RTs

Good Time-sharer IQR RTs

Learning Multitask Skill

Ready to start modeling learning, but what are the phenomena of learning?

Not as well documented as one would like!

Most learning data is either complex tasks, or lacks detail.

In experiments, typically first train single task, then train concurrent performance.

What happens during single-task learning?

E.g. declarative-to-procedural transition.

What happens during dual-task learning?

Combine tasks without modification?

Learn new rules?

Change executive strategy?

Simplest Multiple Task: Overlapping Choice Reaction Tasks

Choice reaction task:

Set of possible stimuli.

Set of possible responses.

A mapping from stimuli to responses.

Observable behavior seems to be at the same grain size as individual production rules.

E.g. S-R rule is close to a single production rule.

Reaction time effects in vicinity of 50 ms period.

Quality data requires:

Explicit costs/payoffs to control speed-accuracy tradeoff.

Rigorous experimental conditions and controls.

Explicit instructions, adequate practice.

Detailed, careful analysis of both group and individual.

Single-task Learning

Actually very little data at this fine-grain level in the literature.

Collecting our own new data; in progress - Steven Lacey.

Some representative single-subject results.

Comparing learning different mappings:

All logically involve only four production rules to map from four stimulus to four responses.

Same responses - high-speed key presses.

Controlled presentation of to-be-learned mapping description.

Surprise: Learning can be extremely rapid!

So rapid that traditional power function does not really fit.

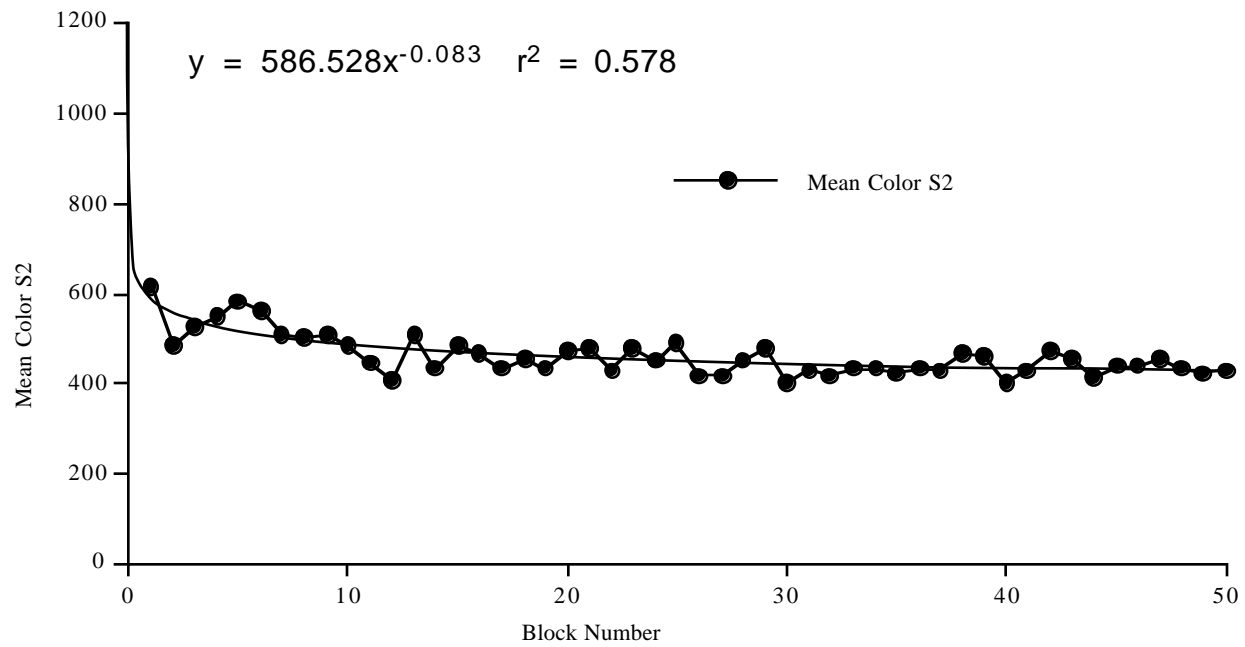
Slower learning shows early semi-plateau, not smooth drop:

Traditional power function misrepresents early phase of learning.

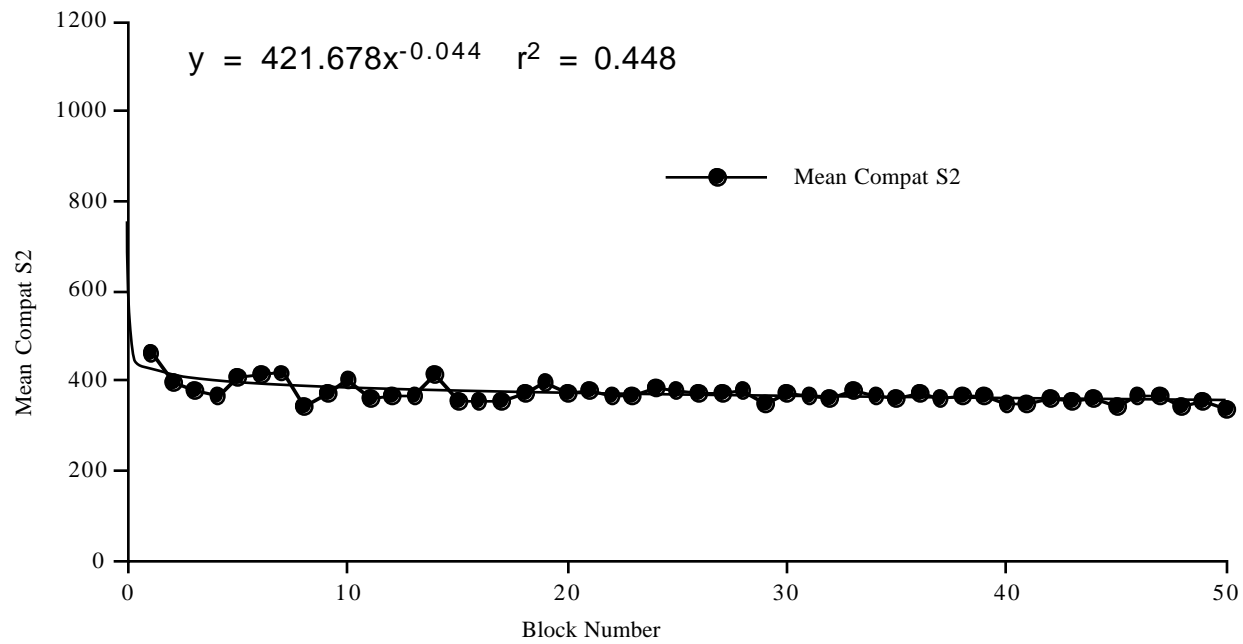
At least one major qualitative shift, possibly more: Collapse of variability after a moderate number of trials.

Aligned with early plateau in RT.

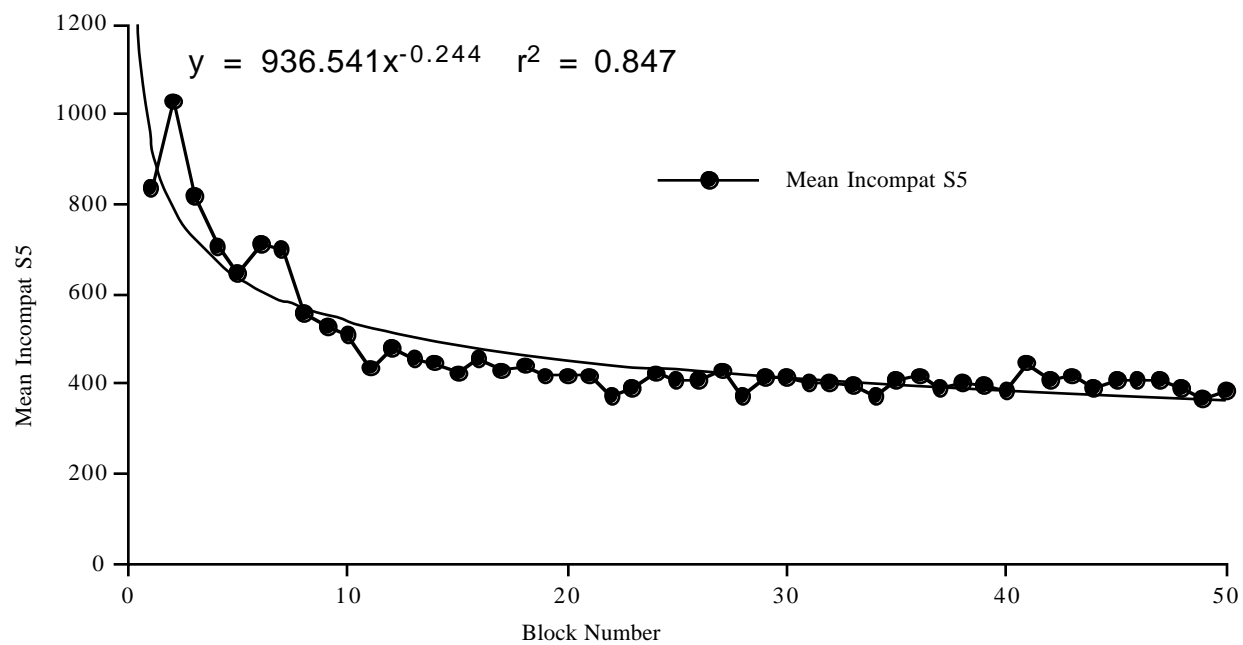
Color-finger mapping Mean RTs



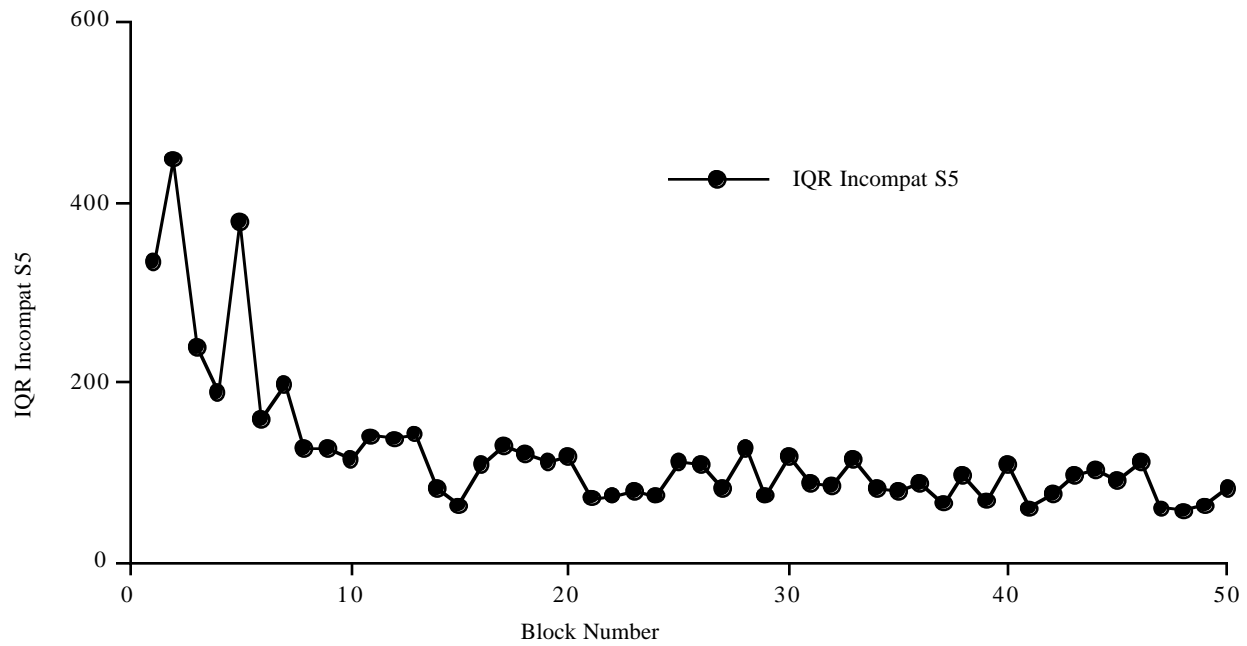
Compatible spatial-finger mapping Mean RTs



Incompatible spatial-finger mapping Mean RTs



Incompatible spatial-finger mapping IQR RTs



Dual-task Learning

Data from our “virtually perfect time-sharing” experiments.

Simultaneous visual-manual and auditory-vocal choice reaction tasks.

Starts in single-task practice, repeated periodically throughout to provide baseline.

Then, dual tasks with two kinds of trials intermixed:

Only one task’s stimulus appears.

- A single task, but in dual-task context.

Both task’s stimuli appear.

- Simultaneous task execution encouraged.

Superficially, same production rules could be used throughout, only coordination requirements change.

Sharp individual differences in ability to execute tasks simultaneously.

Compare a Good and Poor Time-sharer

A single S-R pair from the visual-manual tasks tracked over six sessions.

With and without the concurrent auditory-vocal task.

Good sharer:

Some initial learning for dual task context.

Dual tasks and single-task converge.

Can do the tasks simultaneously with same speed as single.

Burst of variability at each transition.

Poor sharer:

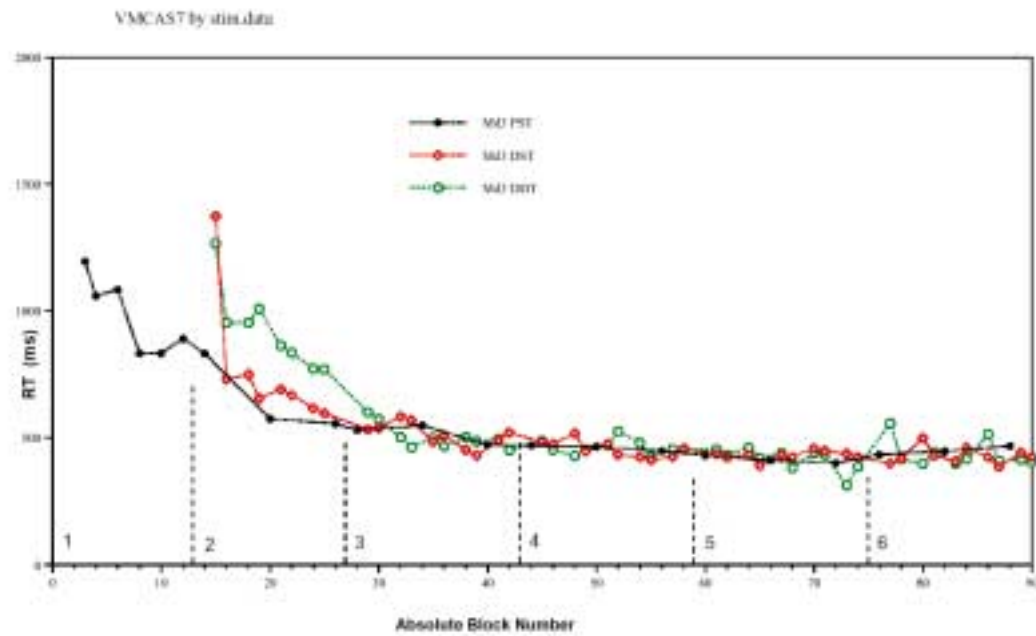
Overall similar speed on single-task to good time-sharer.

Relatively fast at beginning - somewhat flat learning curve.

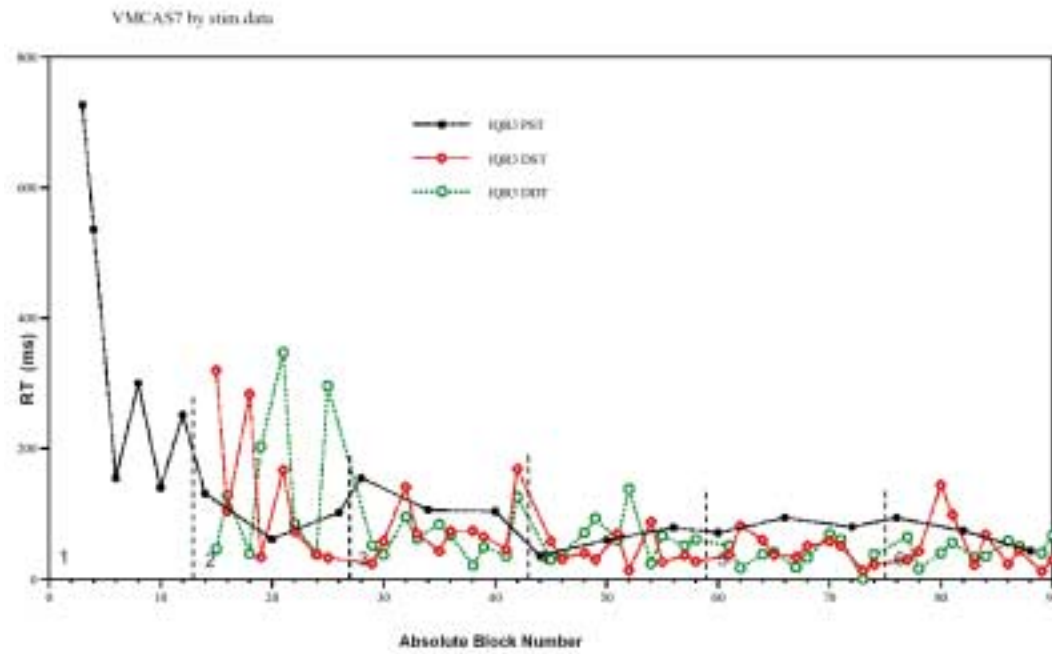
Dual task never quite converges - never “gets” it.

Severe and persistent variability of concurrent task.

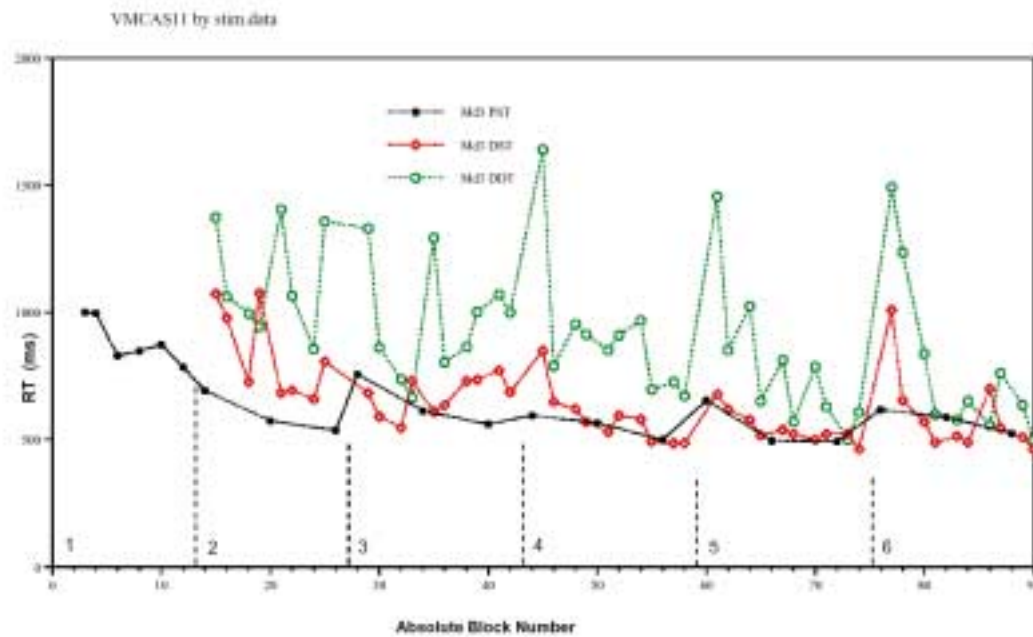
Good Time-sharer Mean RTs



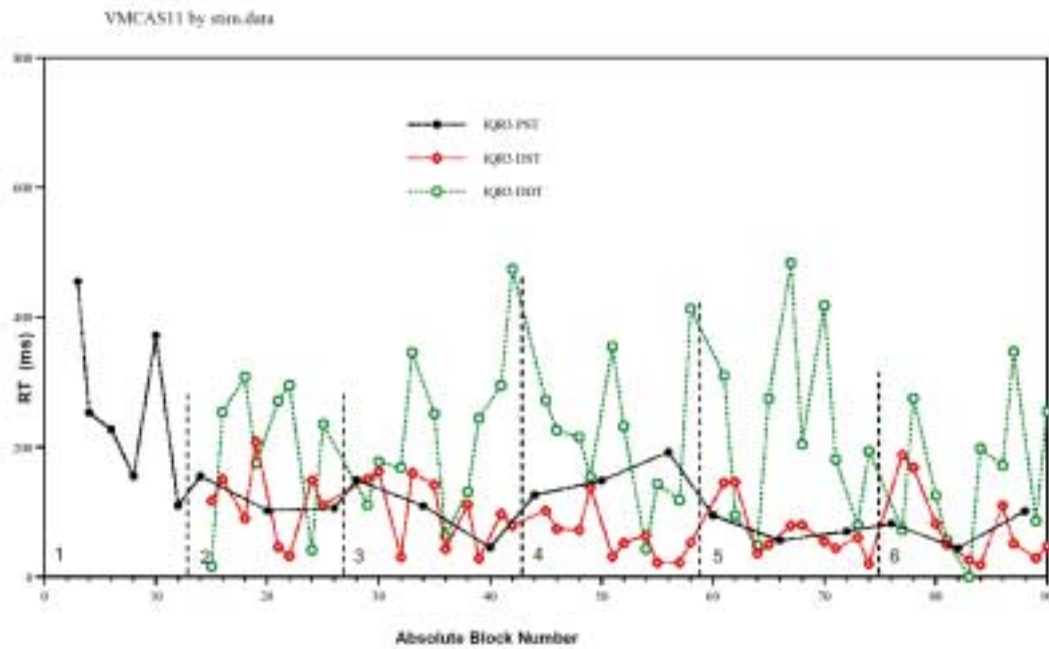
Good Time-sharer IQR RTs



Poor Time-sharer Mean RTs



Poor Time-sharer IQR RTs



Tentative Interpretation

A lot of complexity in learning for such a simple task!

Ability to learn single task is different from ability to learn dual task coordination.

Poor sharer did just fine on single-task, OK on single in dual context, but failed on dual-task.

Suggests that more than a single learning process at work!

Assume a declarative/interpreted stage that can reappear or persist.

Slow-downs, variability associated with acquisition/manipulation of declarative representation.

- Difference in acquisition rates may reflect semantic complexity of declarative representation.
- But stable fast performance does not require that declarative representation is no longer used.
 - Declarative/interpreted models of choice reaction can be just as fast and stable as fully proceduralized models.

Planned Direction for a Learning Mechanism

Question traditional assumptions:

Learning is an automatic subcognitive process that creates new rules that can't be modified, only differentially "strengthened."

Suppose instead:

Most learning phenomena are dependent on interpreted declarative representation that is used for a relatively long time. Cognitive processes (production rules) can:

- Create and modify declarative representations to improve task performance.
- Choose and change executive regime for a task to improve performance or adapt to dual-task requirements.

If so,

Full power of cognitive system is available to perform learning. Don't have to figure out how a "dumb" process can be "smart." Expect many interactions with memory system, learning strategies.

This is a different part of the design space for learning mechanisms in rule-based systems.