Architectures of Intelligence

Lecture 8

Learning from Instructions

university of groningen

$$5 \$ 3 = ?$$

$$5 \$ 3 = 5 + 4 + 3 = 12$$

Three phases of Skill Acquisition

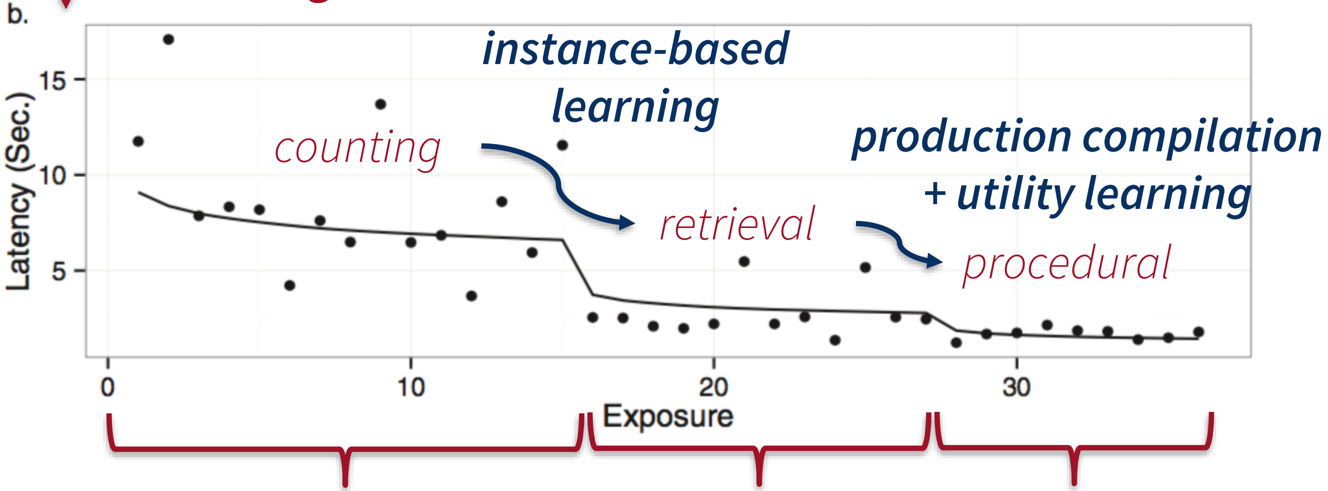
Learning what to do from instructions

5 $ 3 = ?

Cognitive

Associative

Autonomous



instance-based learning

counting

retrieval

production compilation + utility learning

procedural

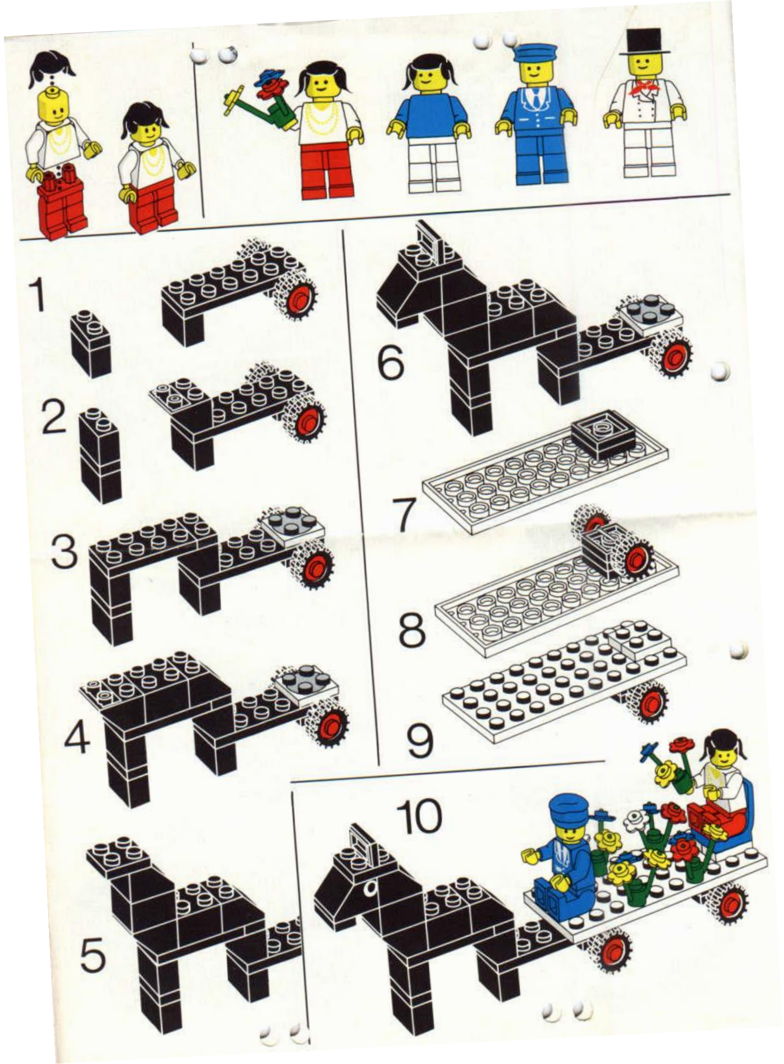
memory activation related to retrieving count facts (5+4=9, 9+3=12)

memory activation related to the answers directly (5$3=12)

no memory activation

Today

- Learning from instructions
- Cognitive Control and the Minimal Control Principle
- Putting it all together:
  Programming the Flight Management System



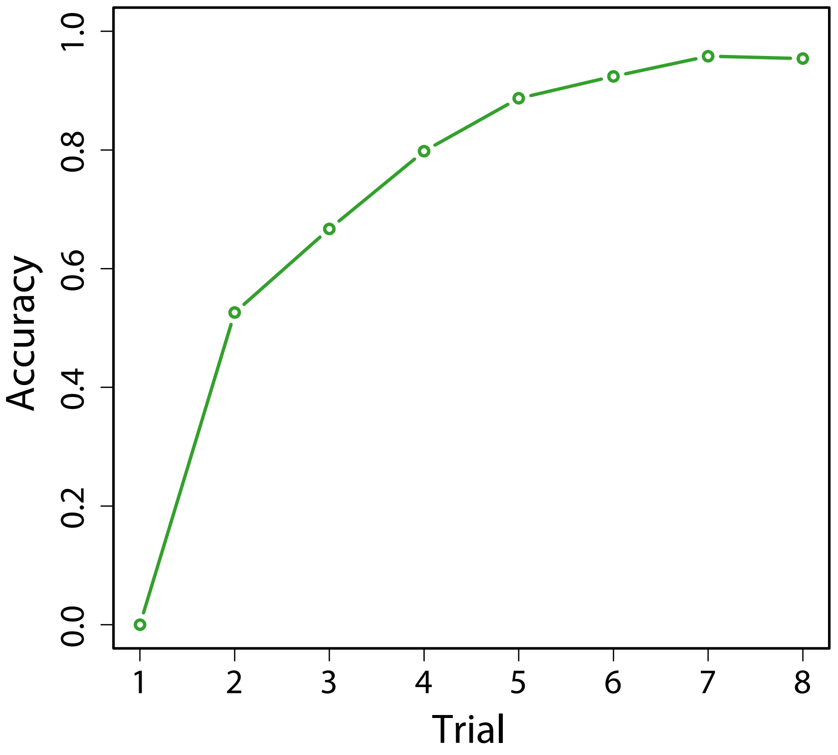
Learning from Instructions

Lecture 5: Paired Associates

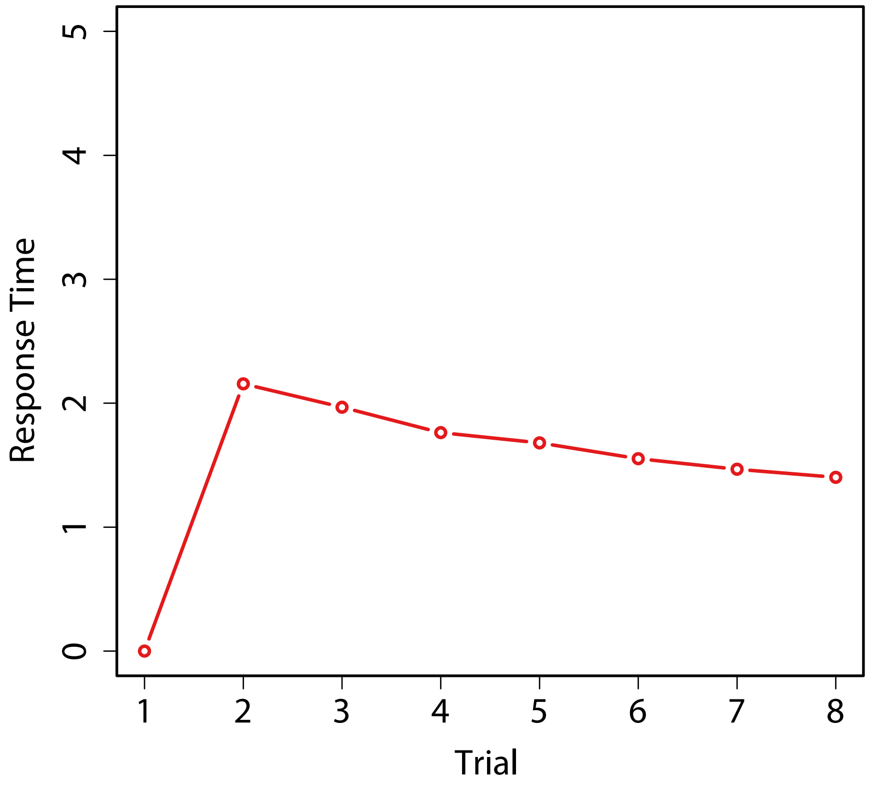
Tent – 2
Zinc – 9
…

Results

- 20 paired associates, 8 repetitions



Accuracy



Response Time

Instructions for paired associates

precondition **action** **postcondition**

- start Read the word stimulus-read
- stimulus-read Retrieve associate recalled
- recalled Test success of recall (response found/wait)
- response found Type response wait
- wait Read feedback new trial
- new trial Complete task start

Instructions for paired associates

precondition **action** **postcondition**

(op1 isa operator pre start action read arg1 fill post stimulus-read)

(op2 isa operator pre stimulus-read action associate arg1 filled arg2 fill post recalled)

(op3 isa operator pre recalled action test-arg2 arg1 respond arg2 wait)

(op4 isa operator pre respond action type arg2 response post wait)

(op5 isa operator pre wait action read arg2 fill post new-trial)

(op6 isa operator pre new-trial action complete-task post start)

Operator productions

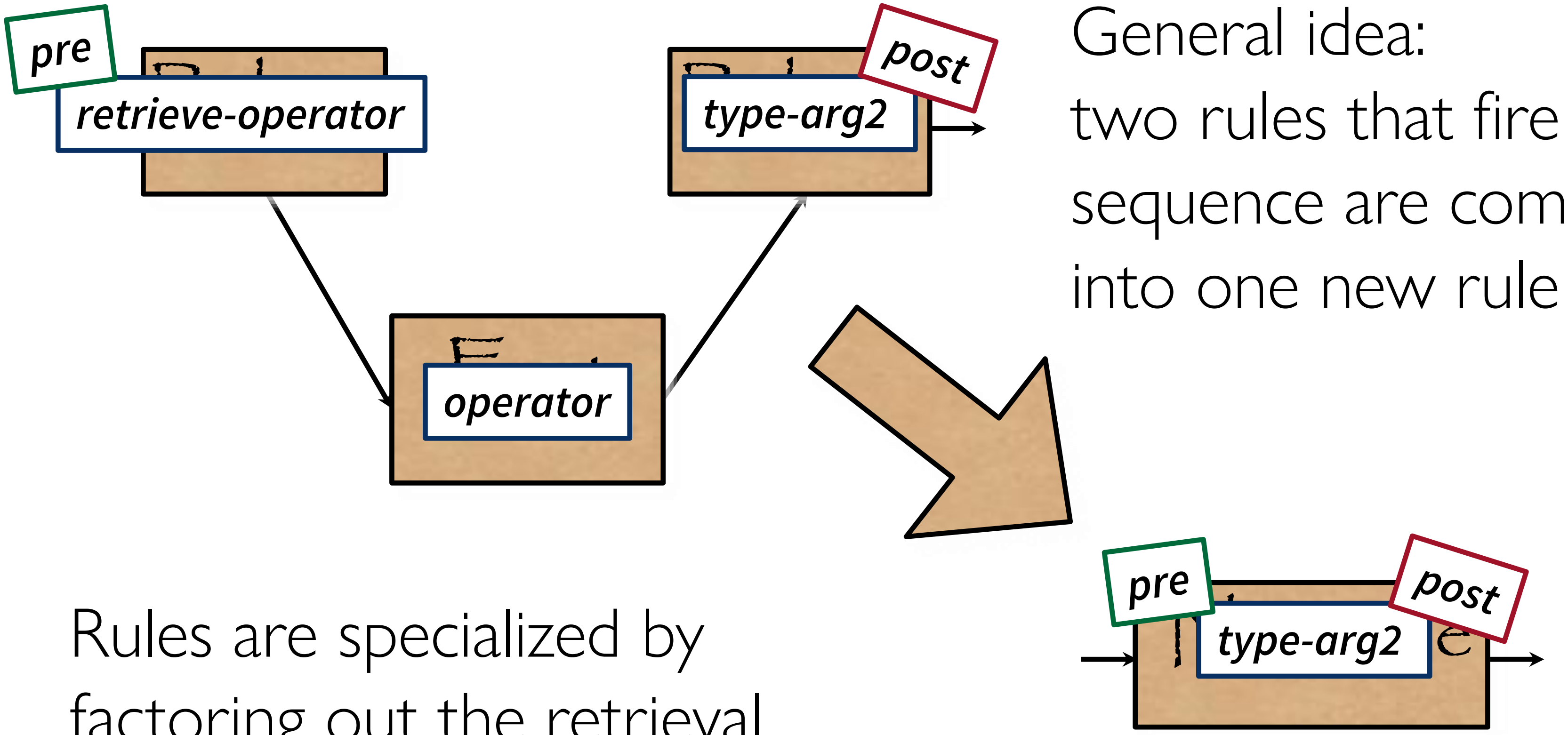
```
(p retrieve-operator
   =goal>
       isa task
       state =state
       step ready
==>
   +retrieval>
       isa operator
        pre =state
    =goal>
       step retrieving-operator)
```

… production compilation …

```
(p type-arg2
   =goal>
       isa task
       step retrieving-operator
   =imaginal>
      isa args
       arg2 =val
   =retrieval>
       isa operator
               type
               sponse
       post =state
   ?manual>
      state    free
==>
    +manual>
      isa      press-key
      key      =val
    =goal>
      state    =state
      step     ready)
```

(op4 isa operator pre respond action type arg2 response post wait)

Production compilation



General idea:
two rules that fire in sequence are combined into one new rule

Rules are specialized by factoring out the retrieval request and retrieval match

Operator productions

```
(p retrieve-operator
   =goal>
       isa task
       state =state
       step ready
==>
   +retrieval>
       isa operator
        pre =state
    =goal>
       step retrieving-operator)
```

```
(p type-arg2
   =goal>
       isa task
       step retrieving-operator
   =imaginal>
      isa args
       arg2 =val
   =retrieval>
       isa operator
       action type
       arg2 response
       post =state
   ?manual>
      state    free
==>
    +manual>
      isa      press-key
      key      =val
    =goal>
      state    =state
      step     ready)
```

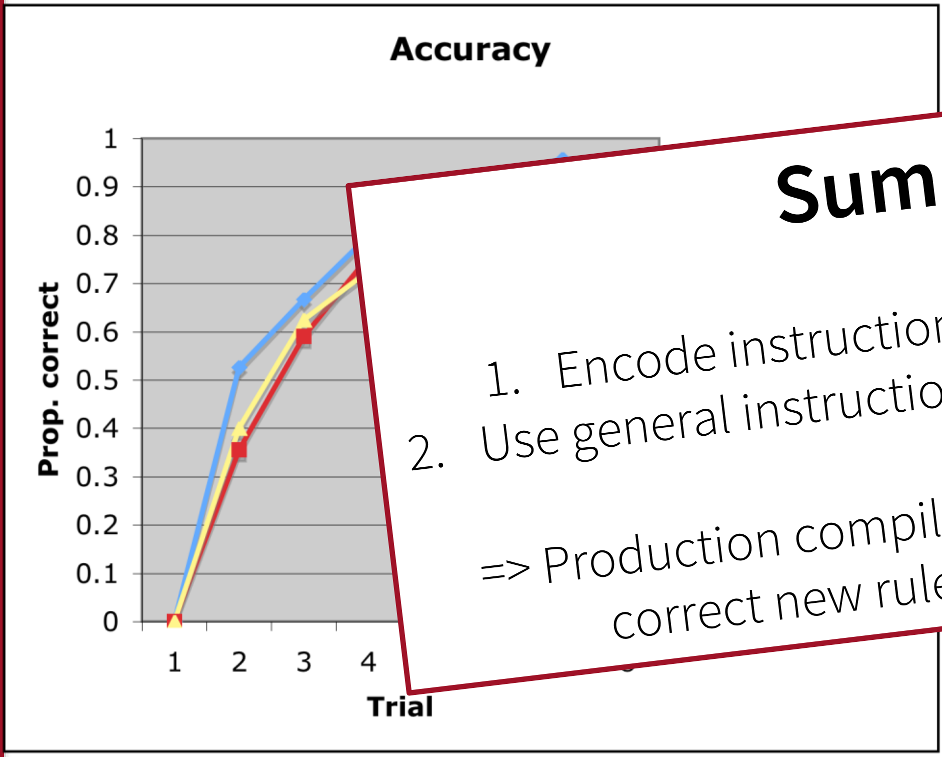
(op4 isa operator pre respond action type arg2 response post wait)

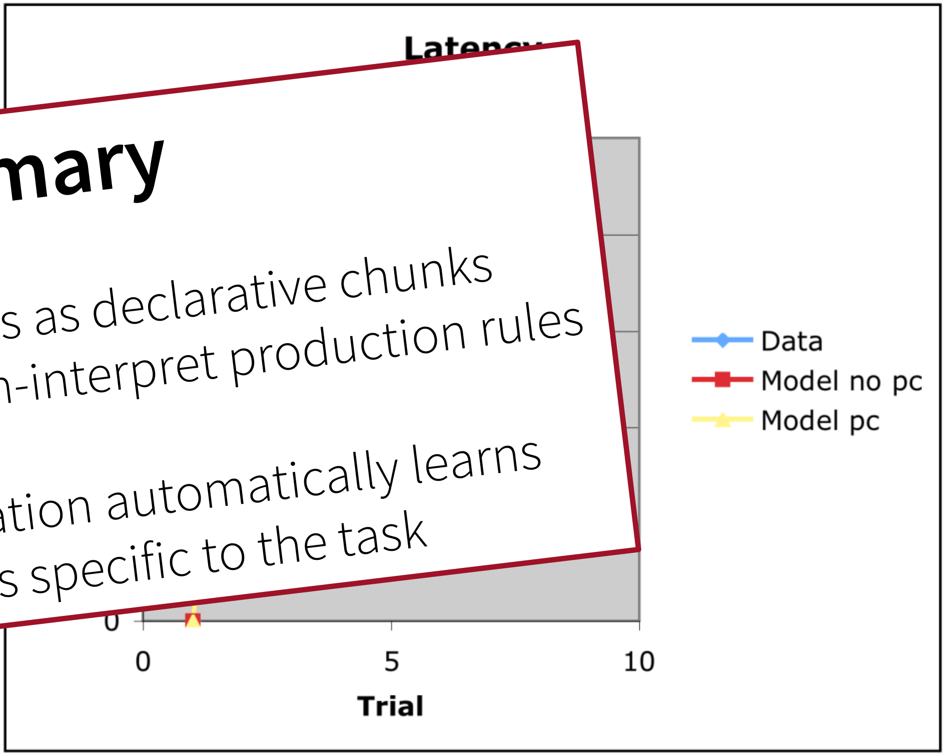
Compiled production

```
(p production323
   =goal>
     isa task
     task assoc
     state respond
     step ready
   =imaginal>
     isa args
     arg2 =val
   ?manual
     state free
==>
   +manual>
     isa press-key
     key =val
    =goal>
       state wait)
```

(op4 isa operator pre respond action type arg2 response post wait)

Model results





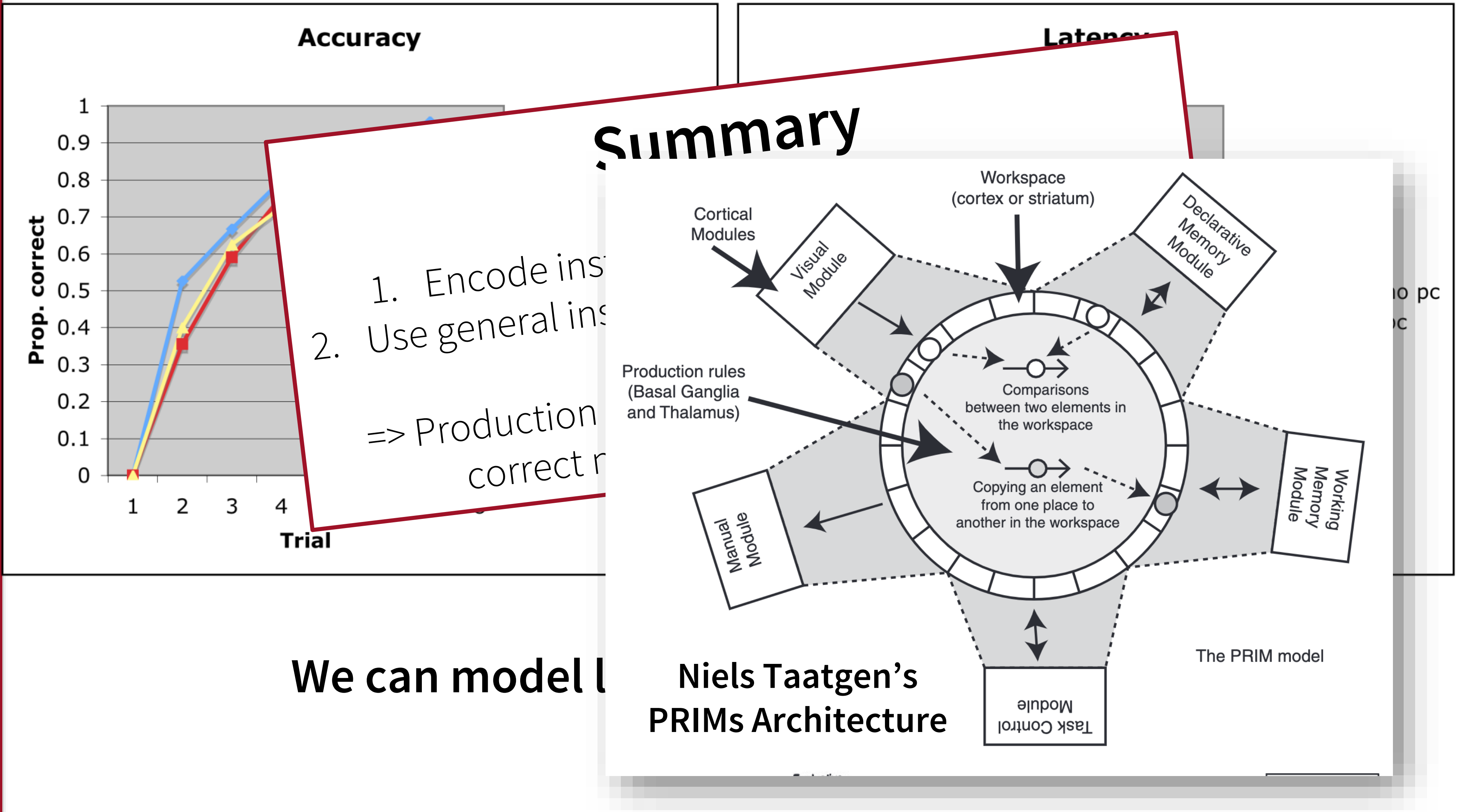
Summary

1. Encode instructions as declarative chunks
2. Use general instruction-interpret production rules

=> Production compilation automatically learns correct new rules specific to the task

We can model learning from instructions!

Model results



Summary

1. Encode ins

2. Use general ins

=> Production

correct r

Niels Taatgen's
PRIMs Architecture

We can model l

Today

- Learning from instructions
- Cognitive Control and the Minimal Control Principle
- Putting it all together:
  Programming the Flight Management System

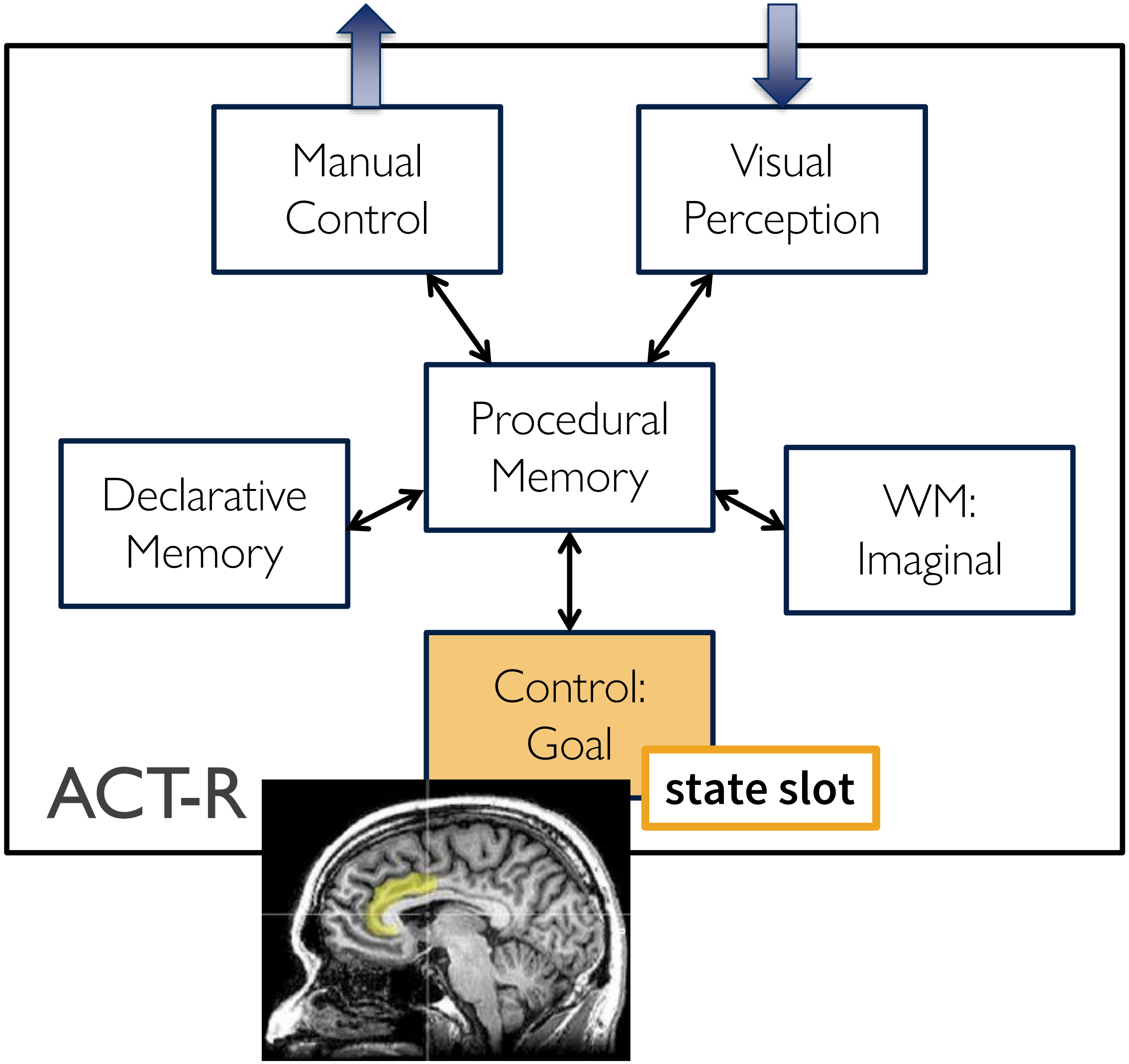
Instructions for paired associate

precondition **action** **postcondition**

- start **Read the word** stimulus-read
- stimulus-read **Retrieve associate** recalled
- recalled **Test success of recall** (response found/wait)
- response found **Type response** wait
- wait **Read feedback** new trial
- new trial **Complete task** start

What is *cognitive control* here?

Cognitive Control in ACT-R

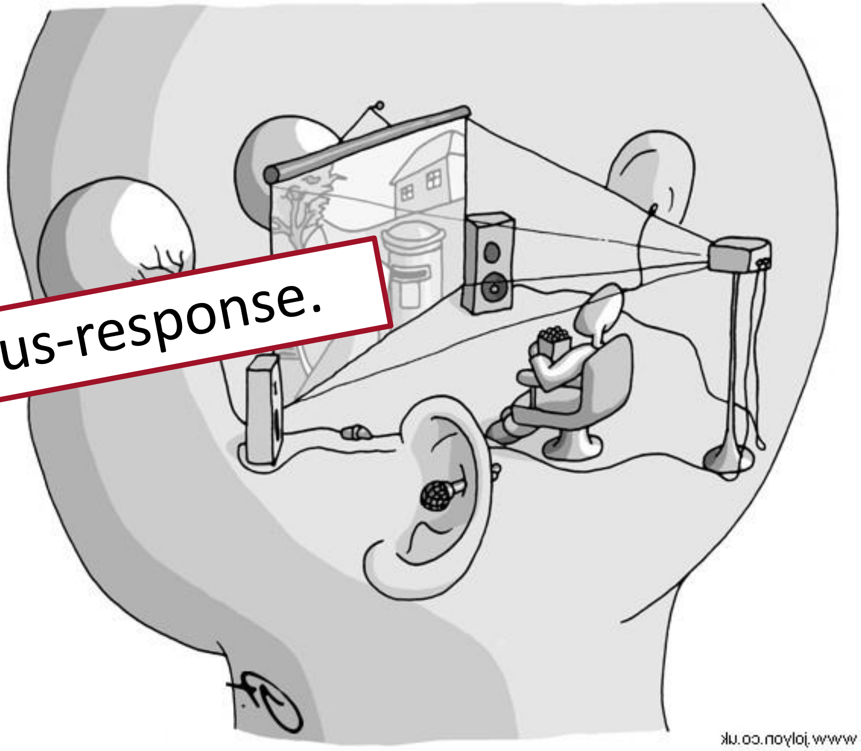


```
(p heads
  =goal>
    ISA      choose
    state    attending
  =visual>
    isa      text
    value    "choose"
  ?manual>
    state    free
==>
  =goal>
    item     "heads"
    state    nil
  +manual>
    ISA      press-key
    key      "H")
```

Cognitive Control in Psychology

- The **executive system** is a theorized cognitive system in psychology that controls and manages other cognitive processes

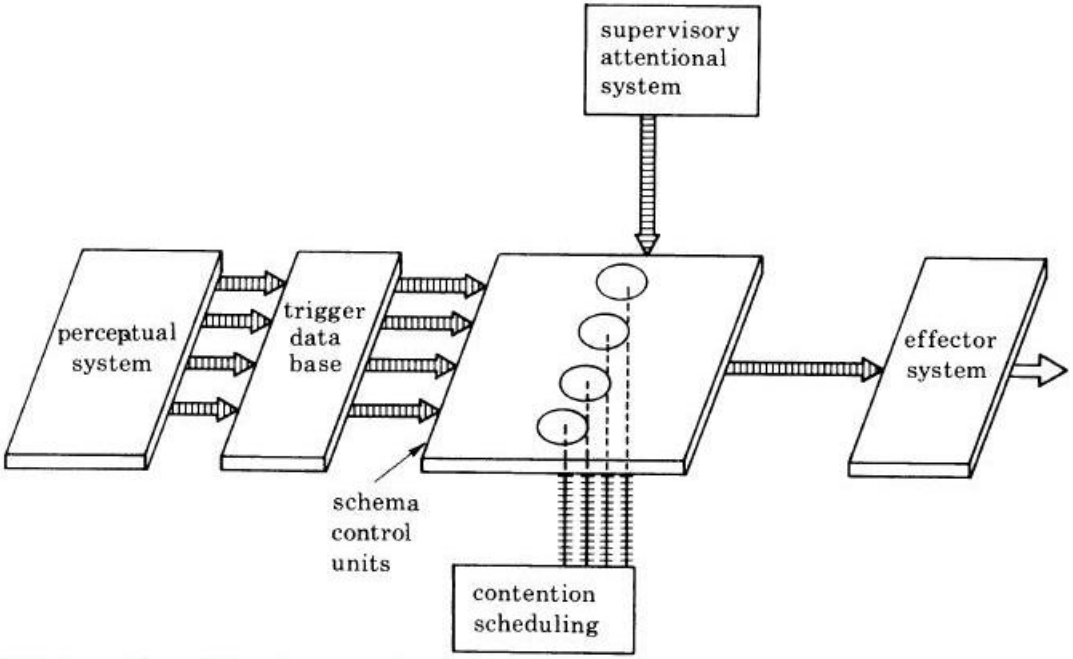
- Norman & Shallice (1986): cognitive control is needed fo
  - that involve planning or decisio
  - that involve error correction or
  - where responses are n novel se
  - angerous or technical
  - that require the overcoming of response or resisting temptatio



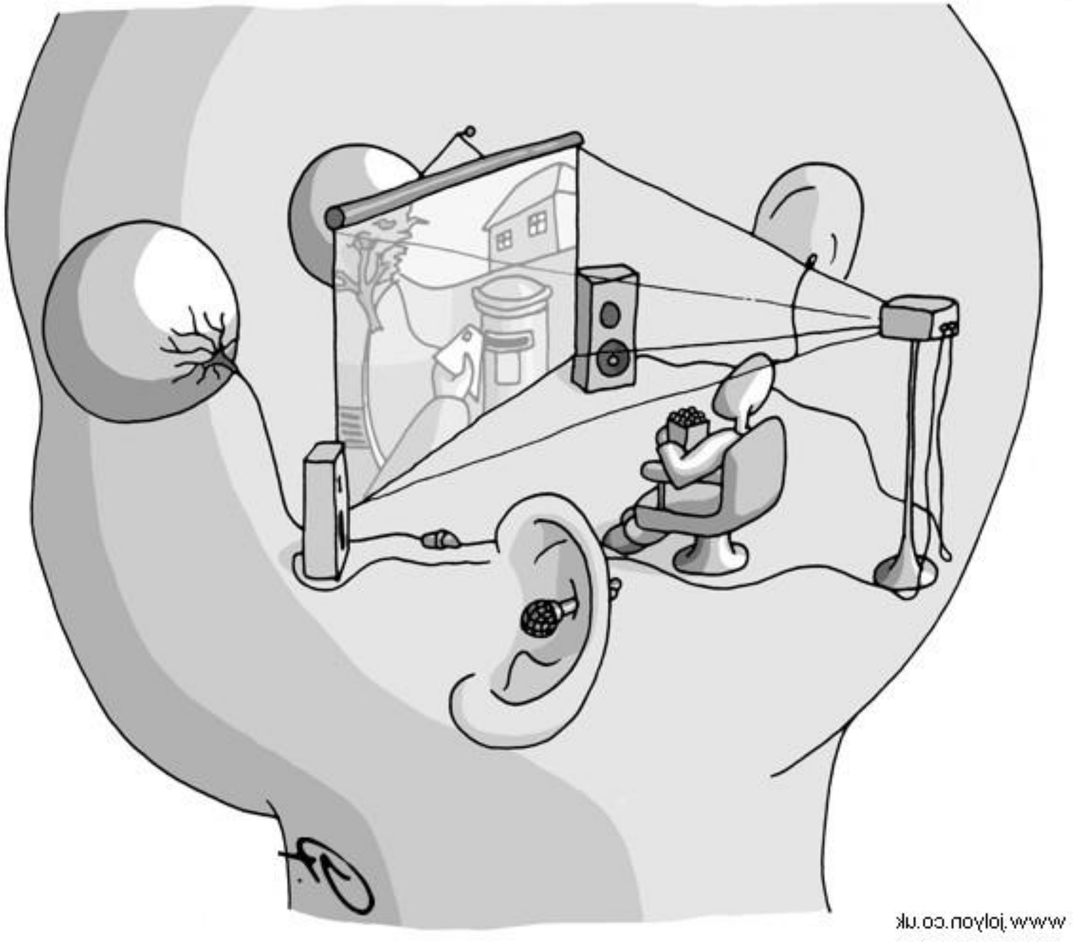
…when cognition is not stimulus-response.

Norman & Shallice (1986)

- Control is in the supervisory attentional system
- Others:
  - Metacognition
  - Central executive
  - etc.

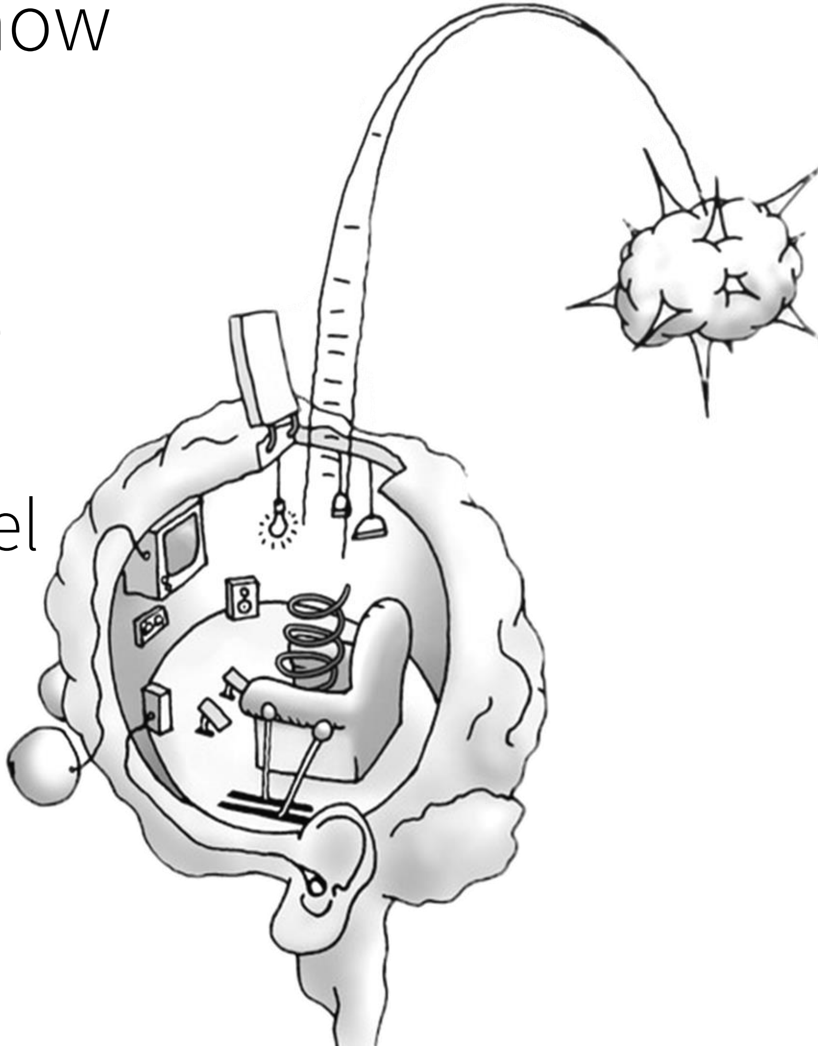


What is the problem of the standard definition of control?

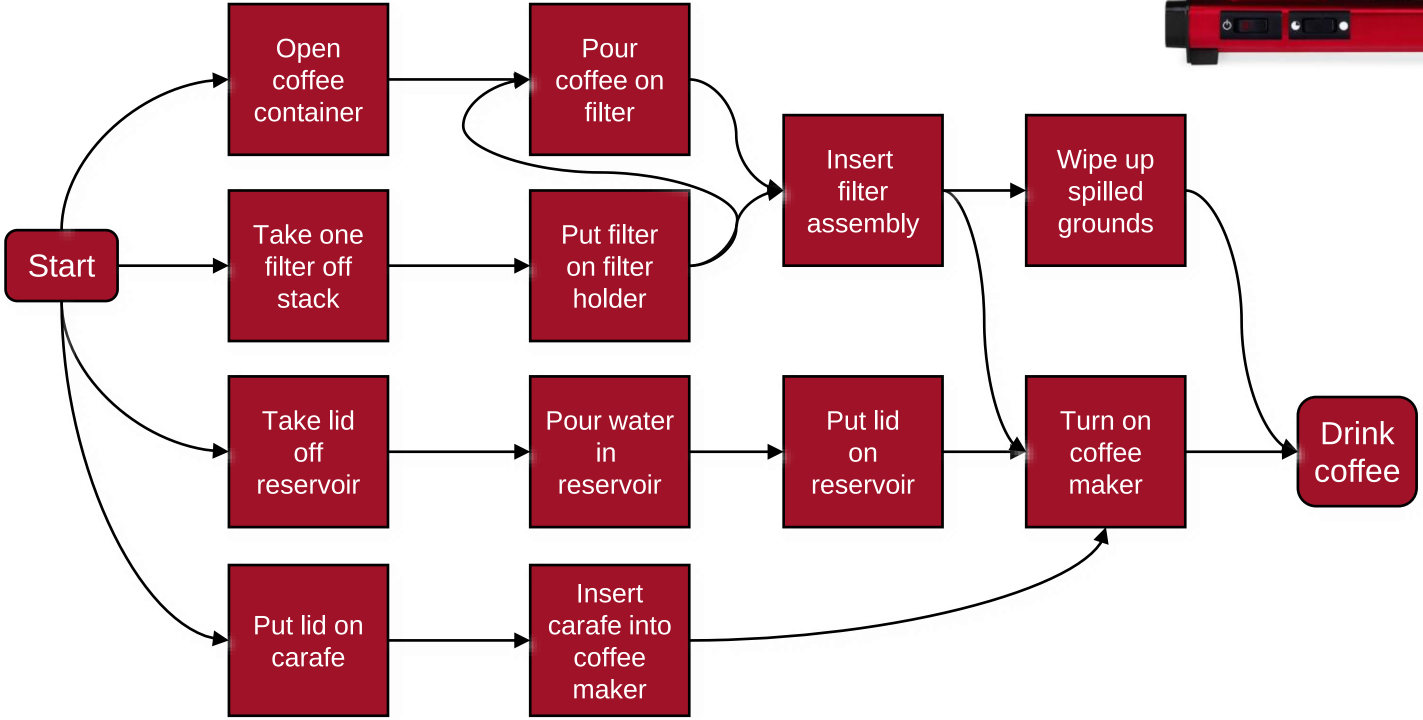


What *is* Cognitive Control?

- Use cognitive models to show **how it could work:**
  1. show how a small set of mechanisms can simulate cognitive control
  2. inspect the resulting model to explain control



How to model: making coffee



Start

Open coffee container

Take one filter off stack

Take lid off reservoir

Put lid on carafe

Pour coffee on filter

Put filter on filter holder

Pour water in reservoir

Insert carafe into coffee maker

Insert filter assembly

Put lid on reservoir

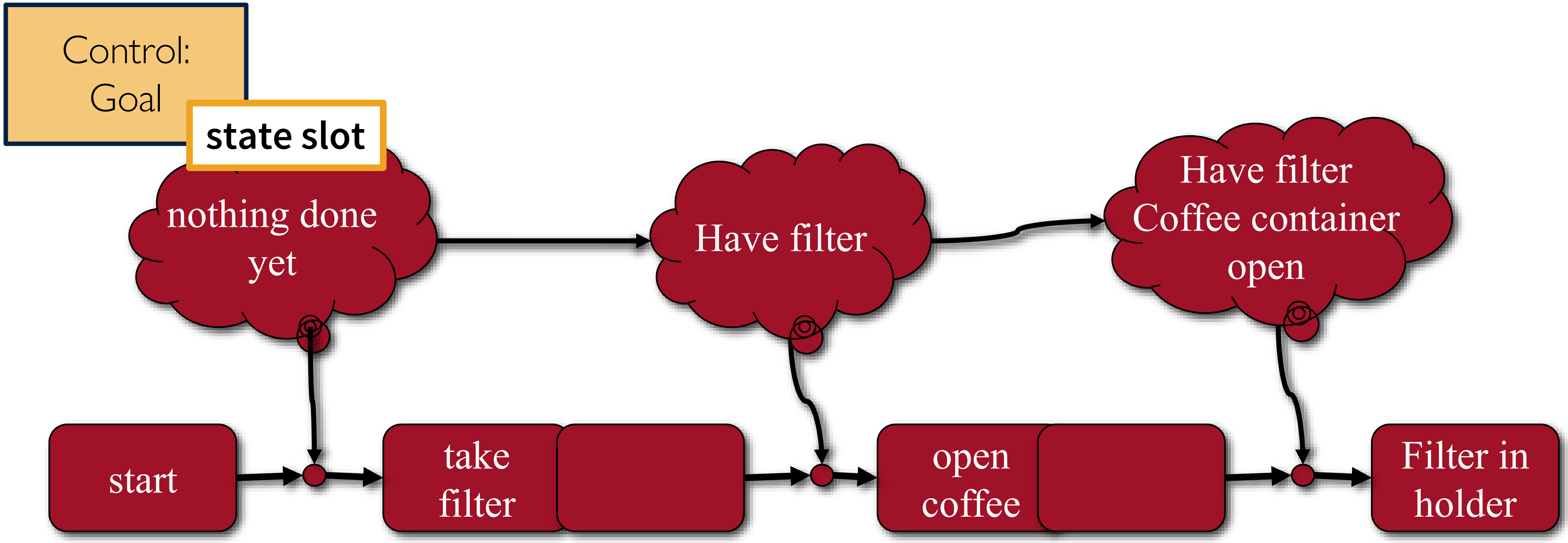
Wipe up spilled grounds

Turn on coffee maker

Drink coffee

Larkin, 1989

Fully controlled coffee brewing



Control:
Goal

state slot

nothing done yet

Have filter

Have filter
Coffee container
open

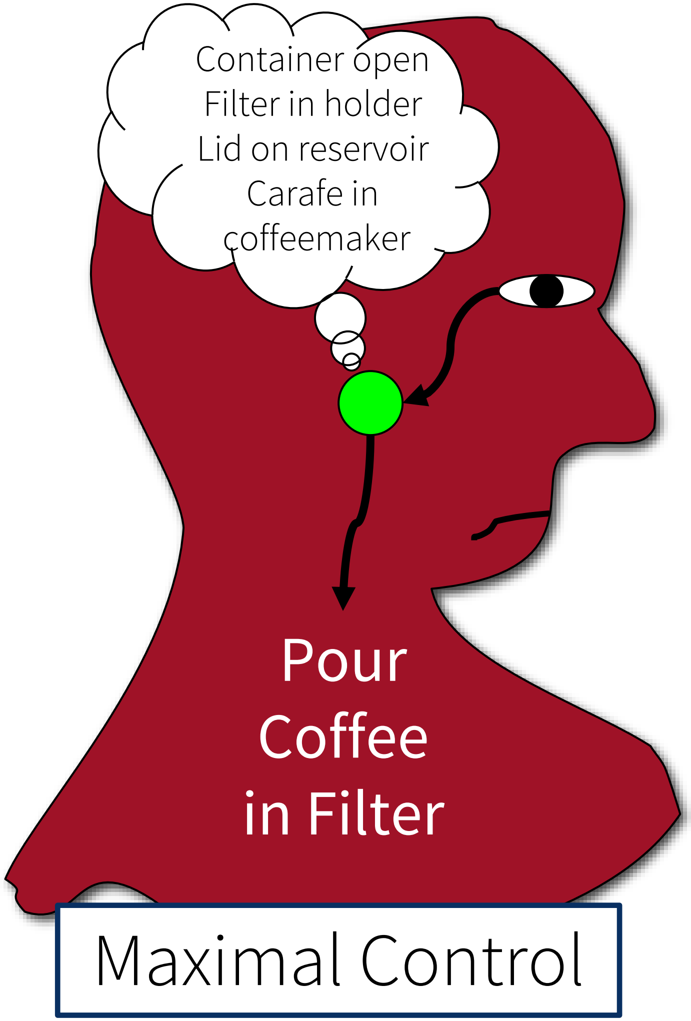
start

take filter

open coffee

Filter in holder

What to do next?

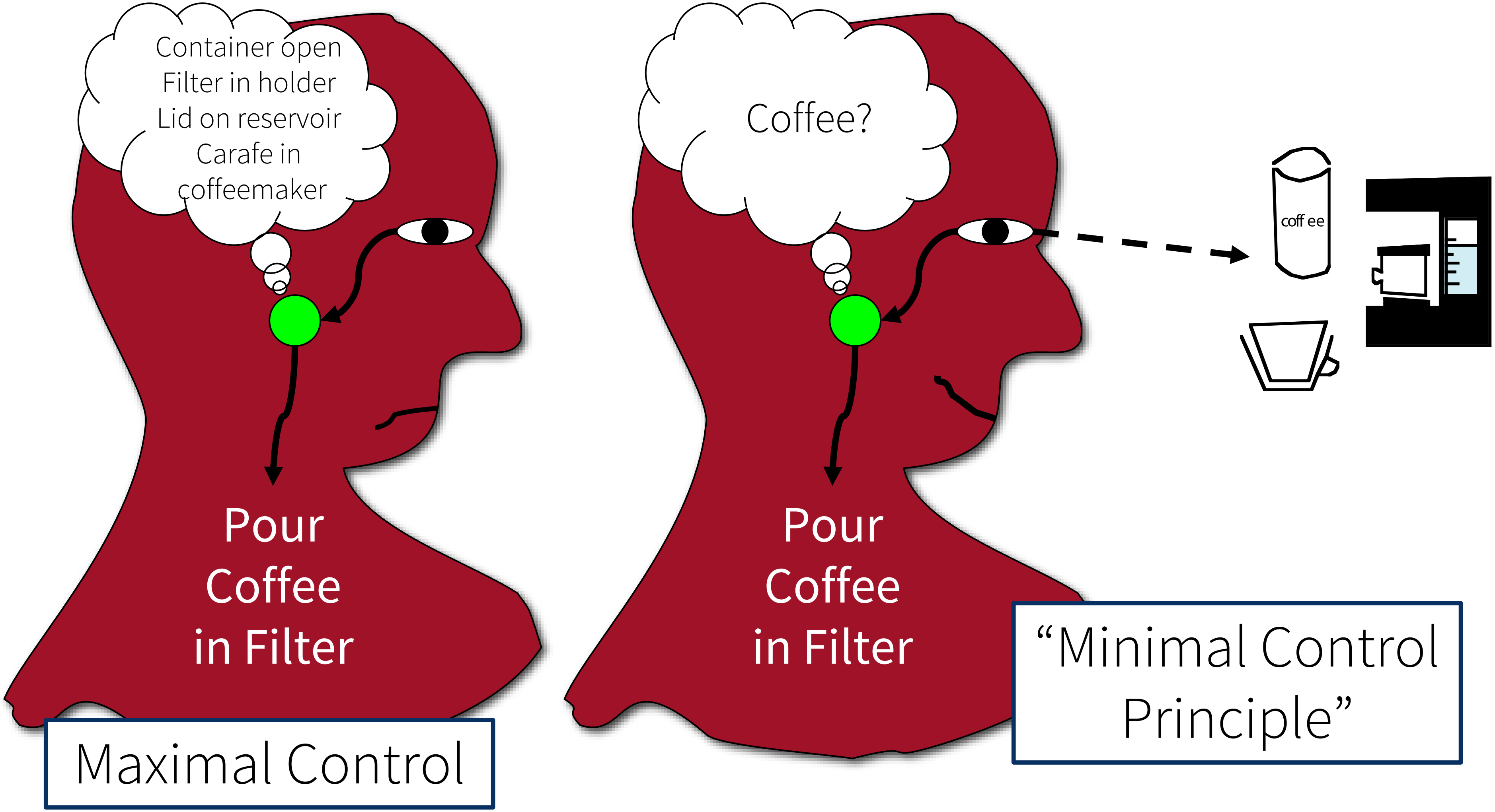


Problems with top-down "maximal control":

- Leads to a cognitive system that has to do a lot of bookkeeping
- Can easily get stuck in a suboptimal sequence
- Brittle behavior

What to do next?

Embedded Cognition



Container open
Filter in holder
Lid on reservoir
Carafe in
coffeemaker

Pour
Coffee
in Filter

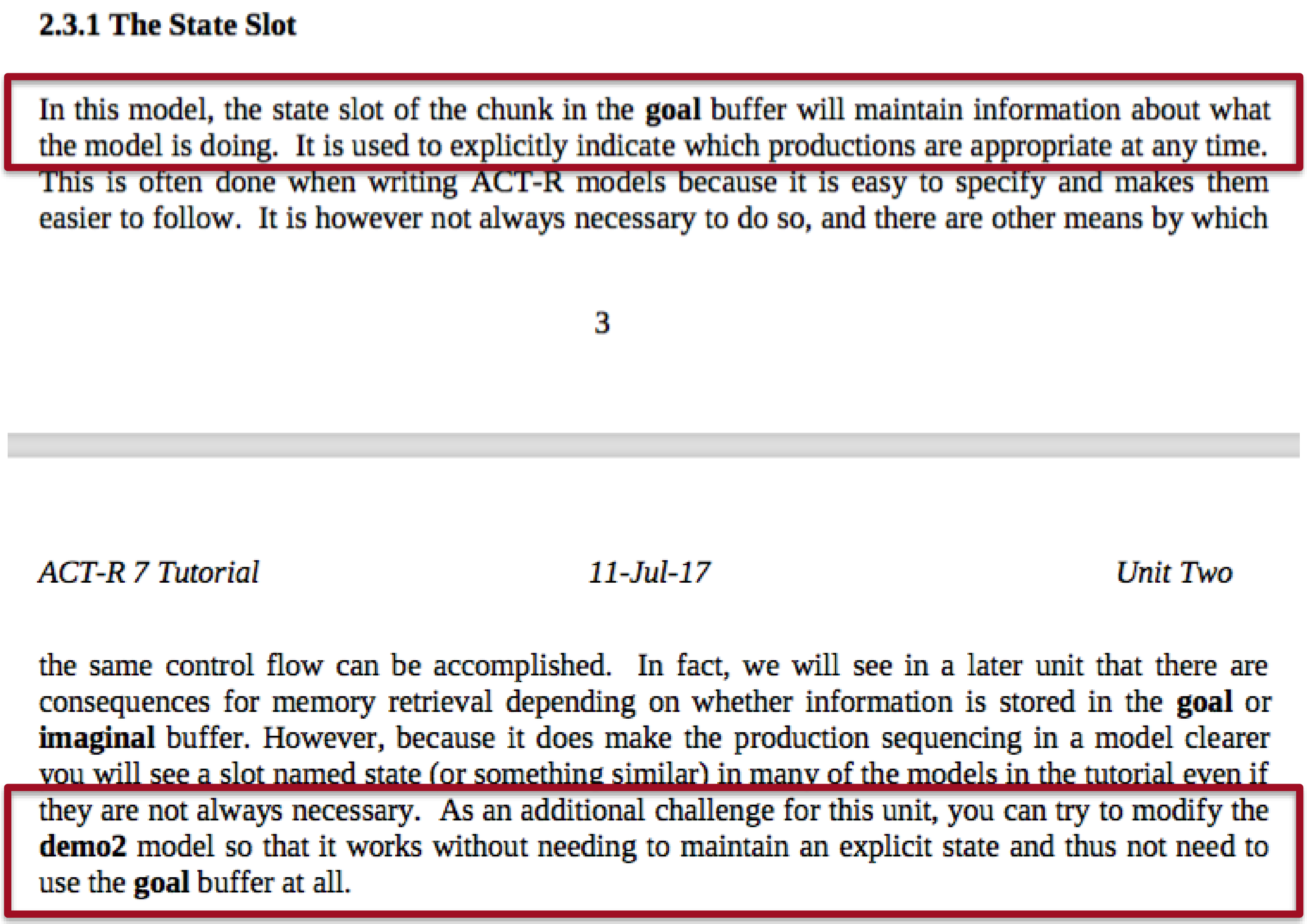
Maximal Control

Coffee?

Pour
Coffee
in Filter

“Minimal Control
Principle”

Unit 2

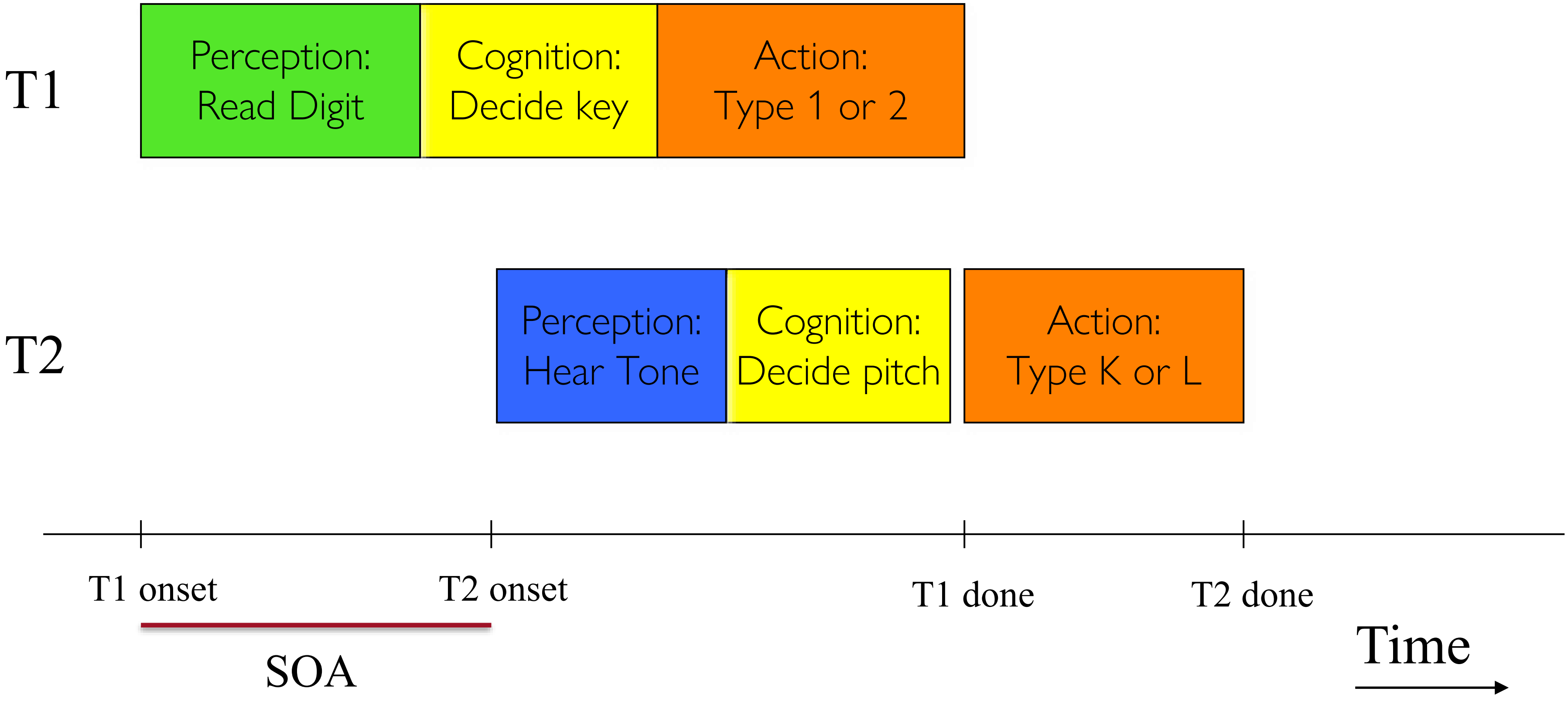


2.3.1 The State Slot

In this model, the state slot of the chunk in the **goal** buffer will maintain information about what the model is doing. It is used to explicitly indicate which productions are appropriate at any time. This is often done when writing ACT-R models because it is easy to specify and makes them easier to follow. It is however not always necessary to do so, and there are other means by which

the same control flow can be accomplished. In fact, we will see in a later unit that there are consequences for memory retrieval depending on whether information is stored in the **goal** or **imaginal** buffer. However, because it does make the production sequencing in a model clearer you will see a slot named state (or something similar) in many of the models in the tutorial even if they are not always necessary. As an additional challenge for this unit, you can try to modify the **demo2** model so that it works without needing to maintain an explicit state and thus not need to use the **goal** buffer at all.

Modeling Control: PRP



Modeling Control: PRP

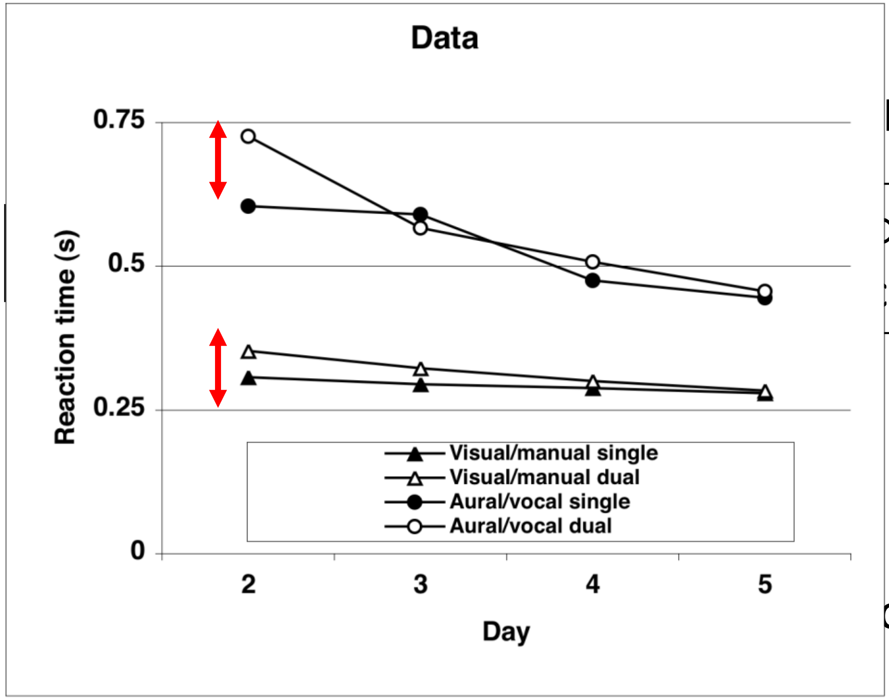
Task 1: Visual

o--

-o-

index finger

middle finger



ral Vocal

dle
h

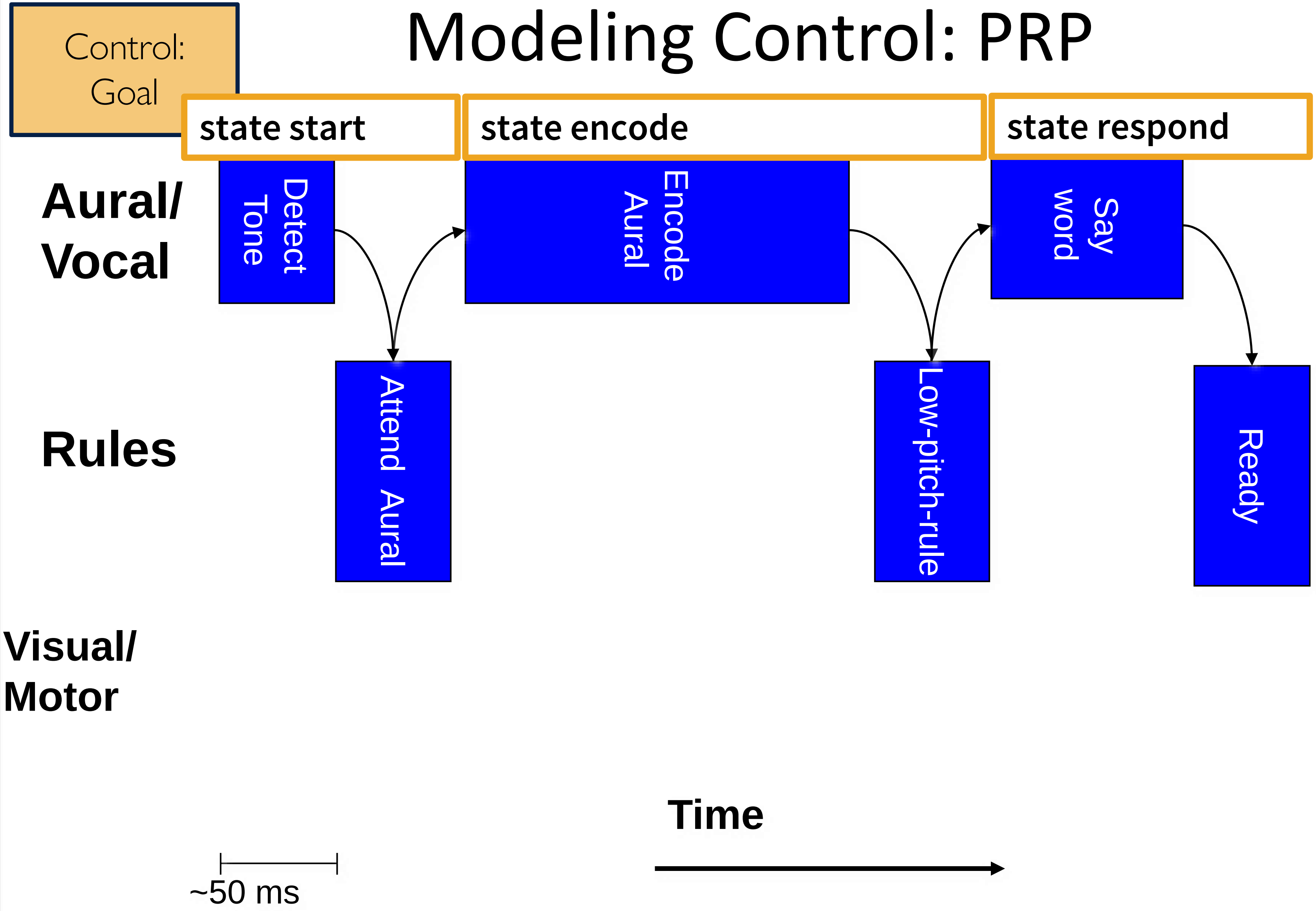
high pitch

o”

“three”

Initially, people have significant dual-tasking interference, but with practice they achieve perfect time-sharing

Modeling Control: PRP



Control: Goal

state start

state encode

state respond

Aural/ Vocal

Rules

Visual/ Motor

Detect Tone

Attend Aural

Encode Aural

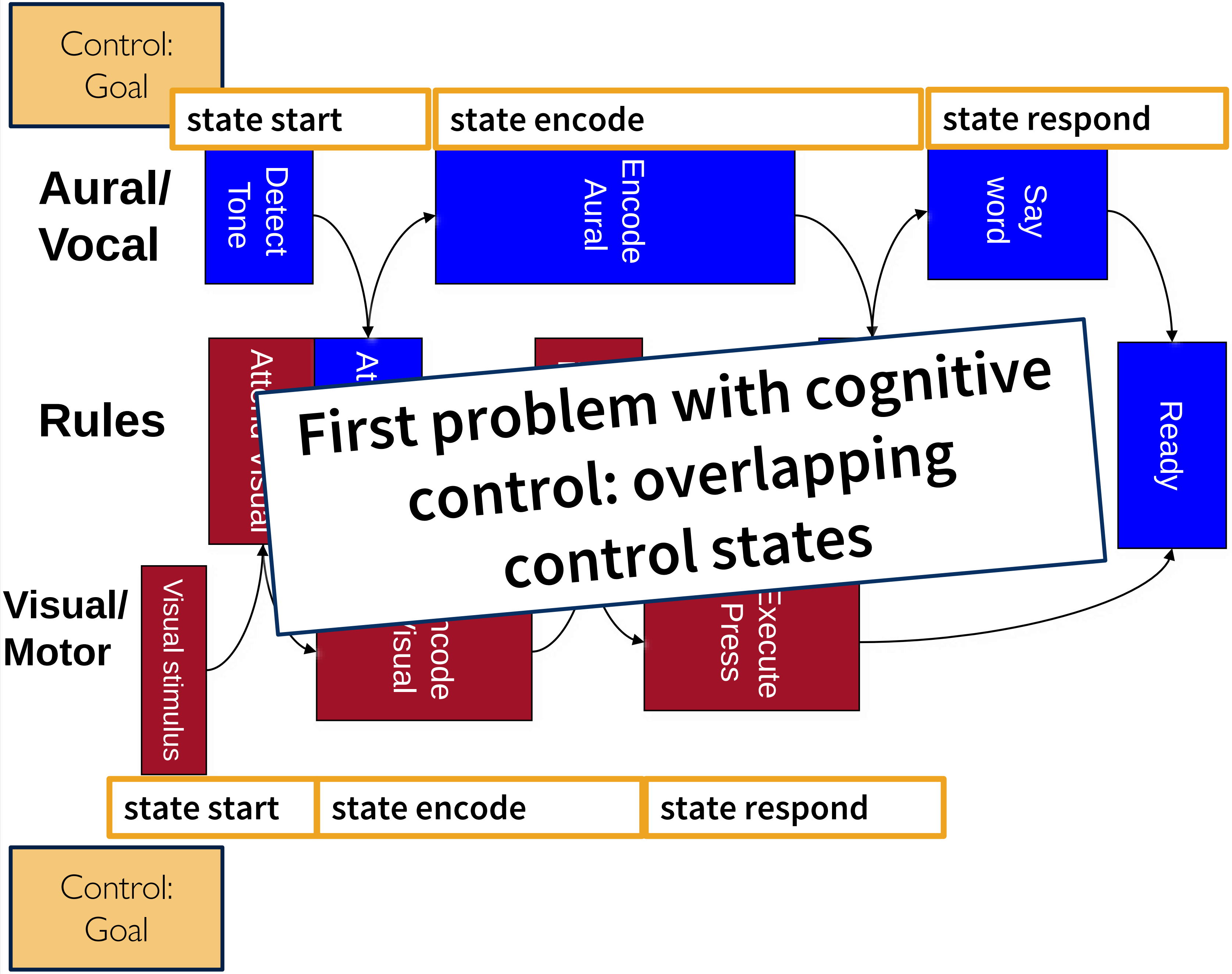
Low-pitch-rule

Say word

Ready

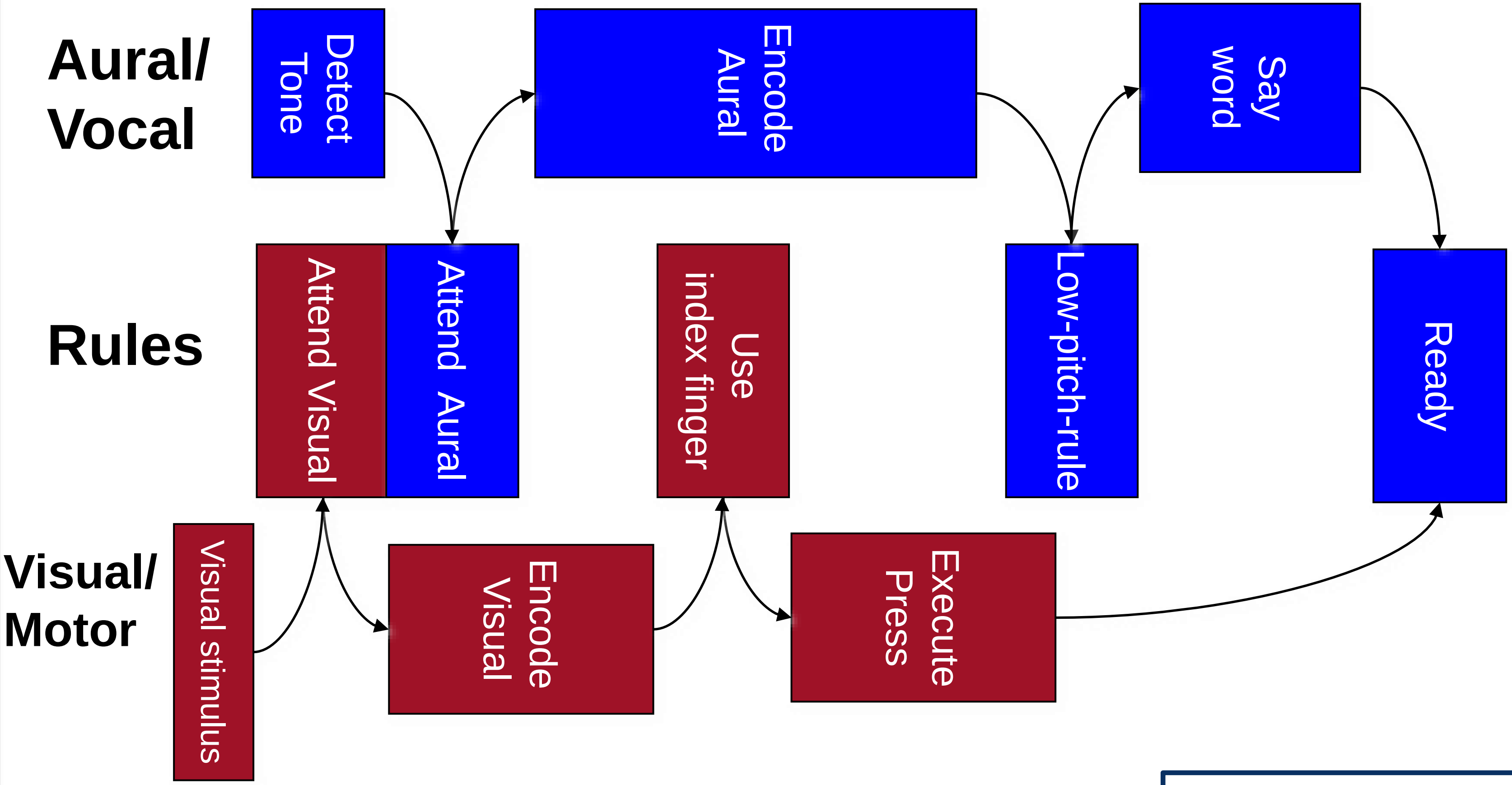
Time

~50 ms



First problem with cognitive control: overlapping control states

Solution: react to environment



Aural/ Vocal

Detect Tone

Encode Aural

Say word

Rules

Attend Visual

Attend Aural

Use index finger

Low-pitch-rule

Ready

Visual/ Motor

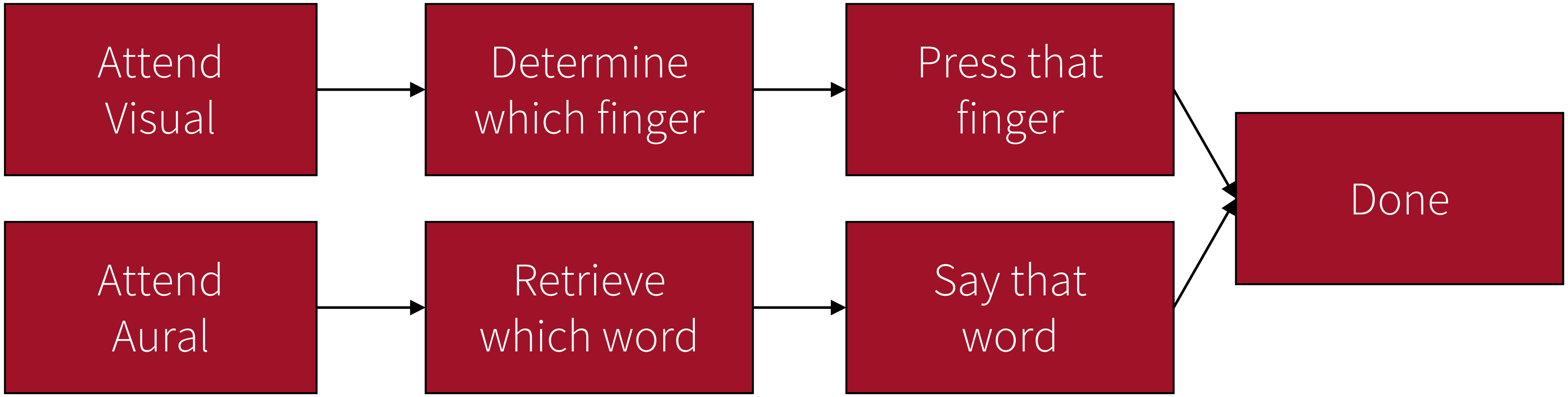
Visual stimulus

Encode Visual

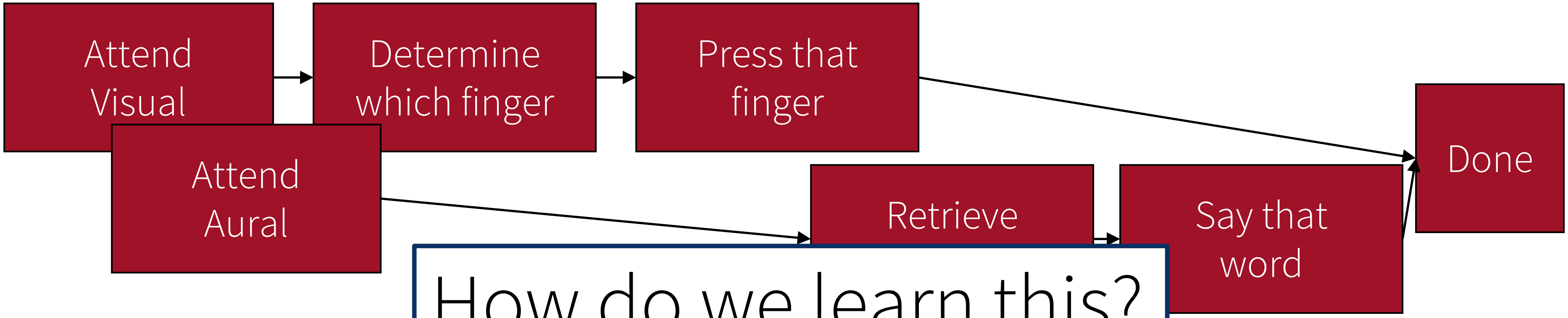
Execute Press

“Minimal Control Principle”

Second problem with predefined cognitive control: How to order the steps?



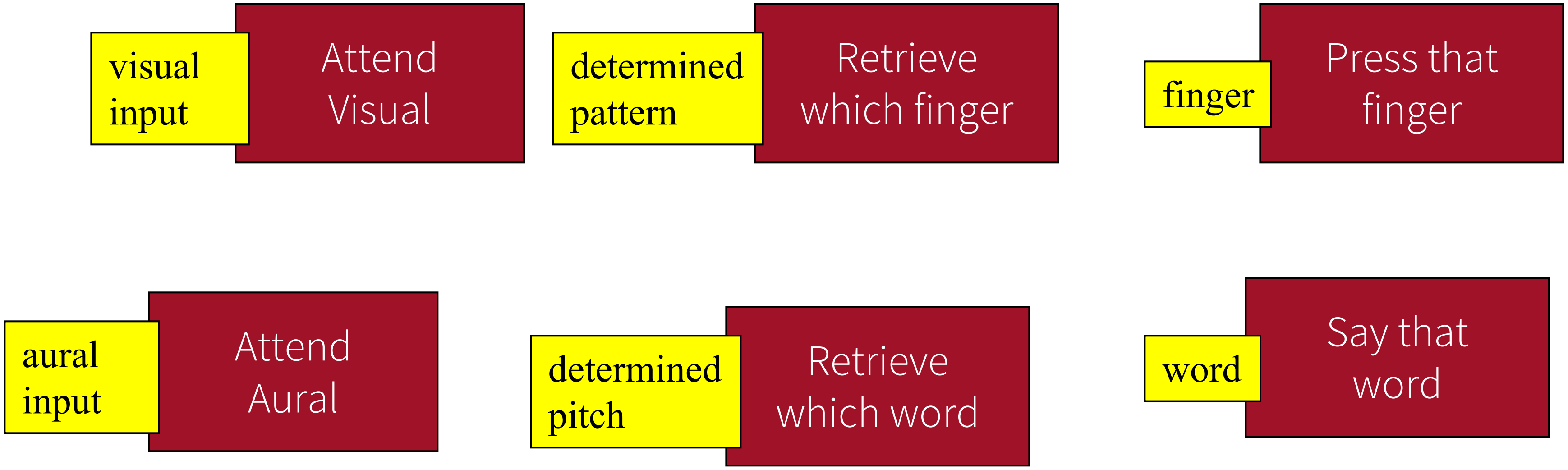
…can be ordered in 45 different ways, but the only one that avoids all dual-task costs is:



How do we learn this?

Learning from instructions

Task instruction in Declarative Memory:



visual input — Attend Visual

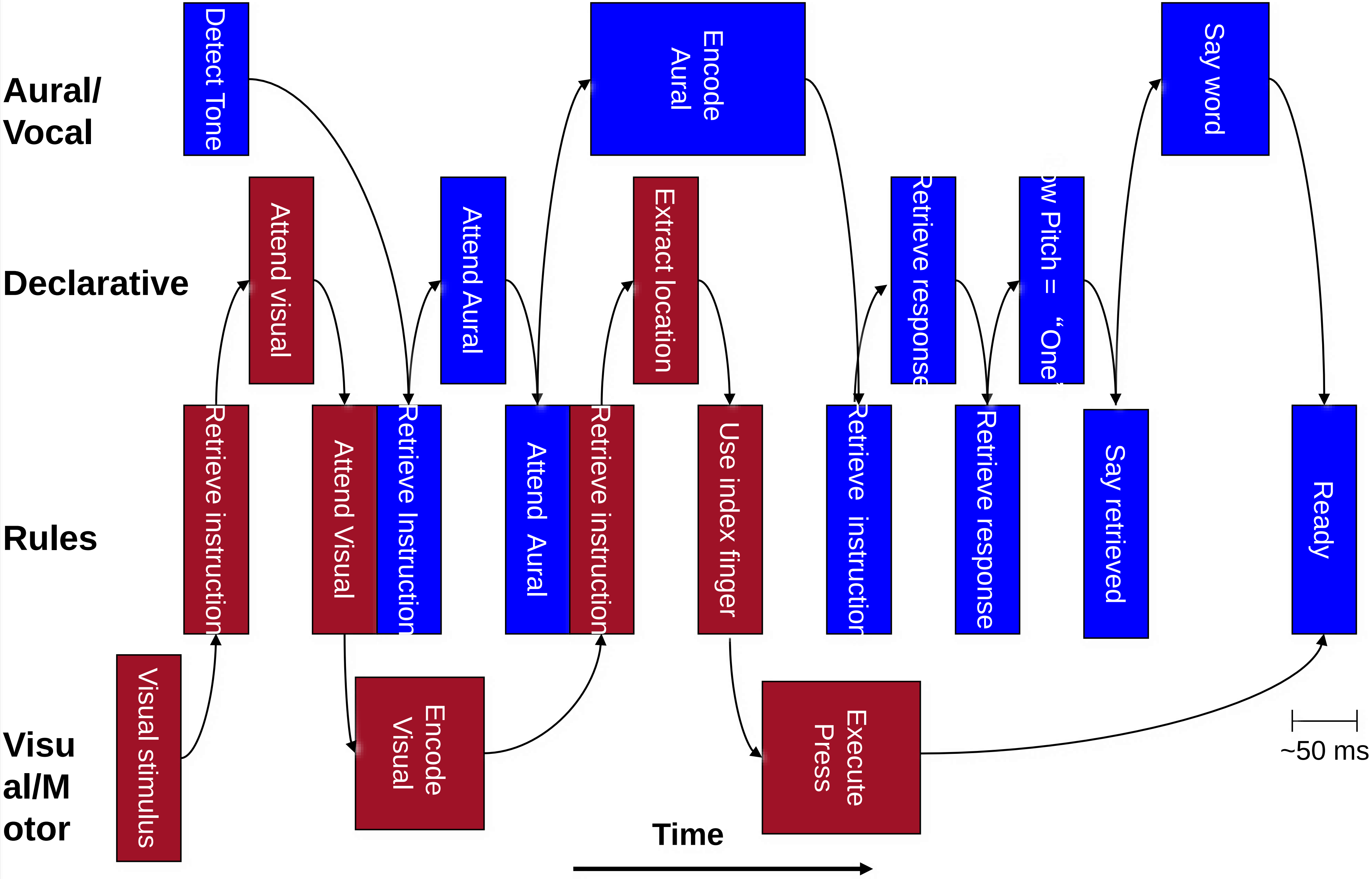
determined pattern — Retrieve which finger

finger — Press that finger

aural input — Attend Aural

determined pitch — Retrieve which word

word — Say that word



Aural/ Vocal

Declarative

Rules

Visu al/M otor

Visual stimulus

Retrieve instruction

Detect Tone

Attend visual

Attend Visual

Retrieve Instruction

Encode Visual

Attend Aural

Attend Aural

Retrieve instruction

Encode Aural

Extract location

Use index finger

Execute Press

Retrieve instruction

Retrieve response

Retrieve response

ow Pitch = “One

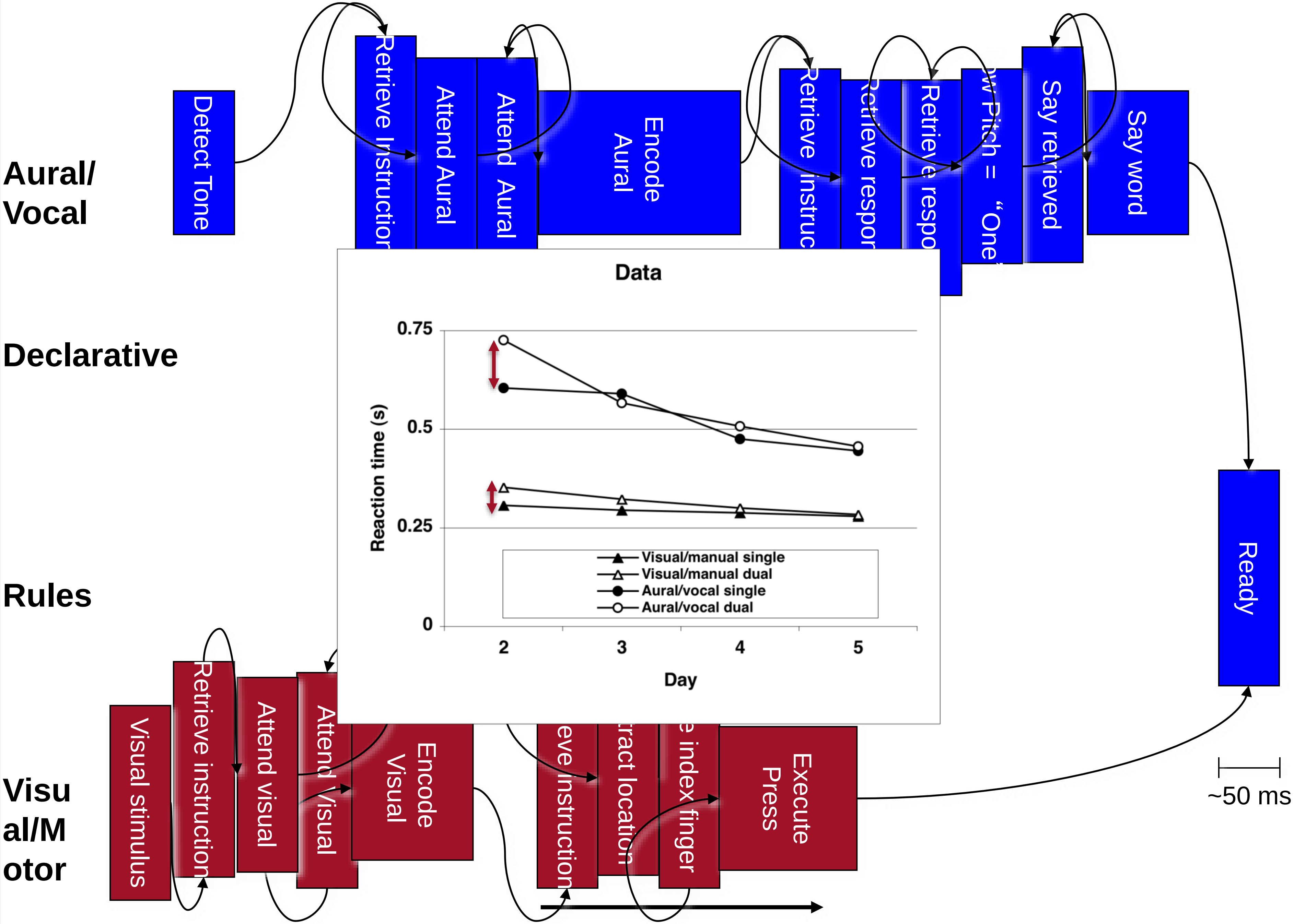
Say retrieved

Say word

Ready

~50 ms

Time



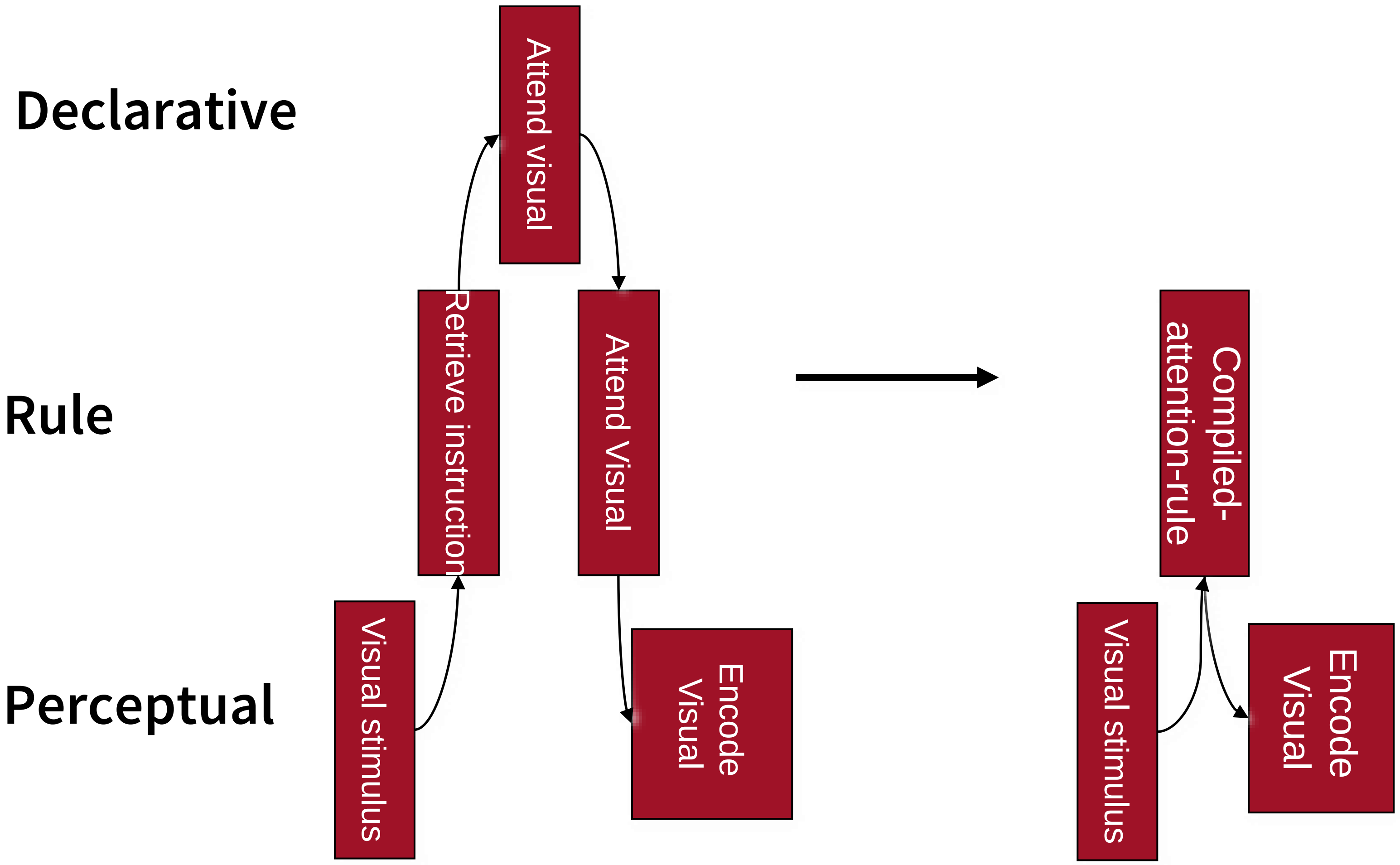
Aural/ Vocal

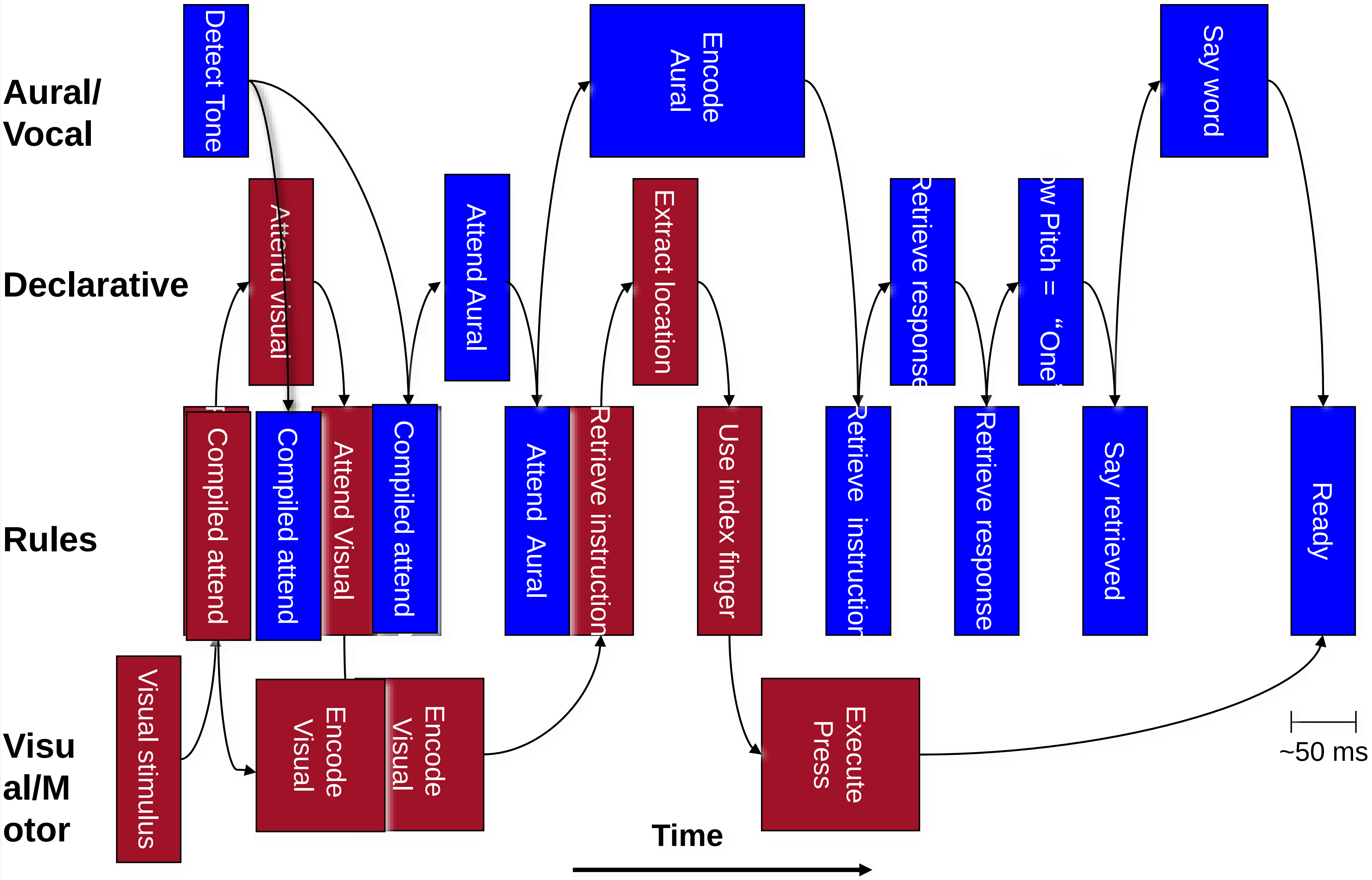
Declarative

Rules

Visu al/M otor

From instructions to productions: Production compilation





Aural/ Vocal

Detect Tone

Encode Aural

Say word

Declarative

Attend visual

Attend Aural

Extract location

Retrieve response

ow Pitch = “One

Rules

Compiled attend

Compiled attend

Attend Visual

Compiled attend

Attend Aural

Retrieve instruction

Use index finger

Retrieve instruction

Retrieve response

Say retrieved

Ready

Visu al/M otor

Visual stimulus

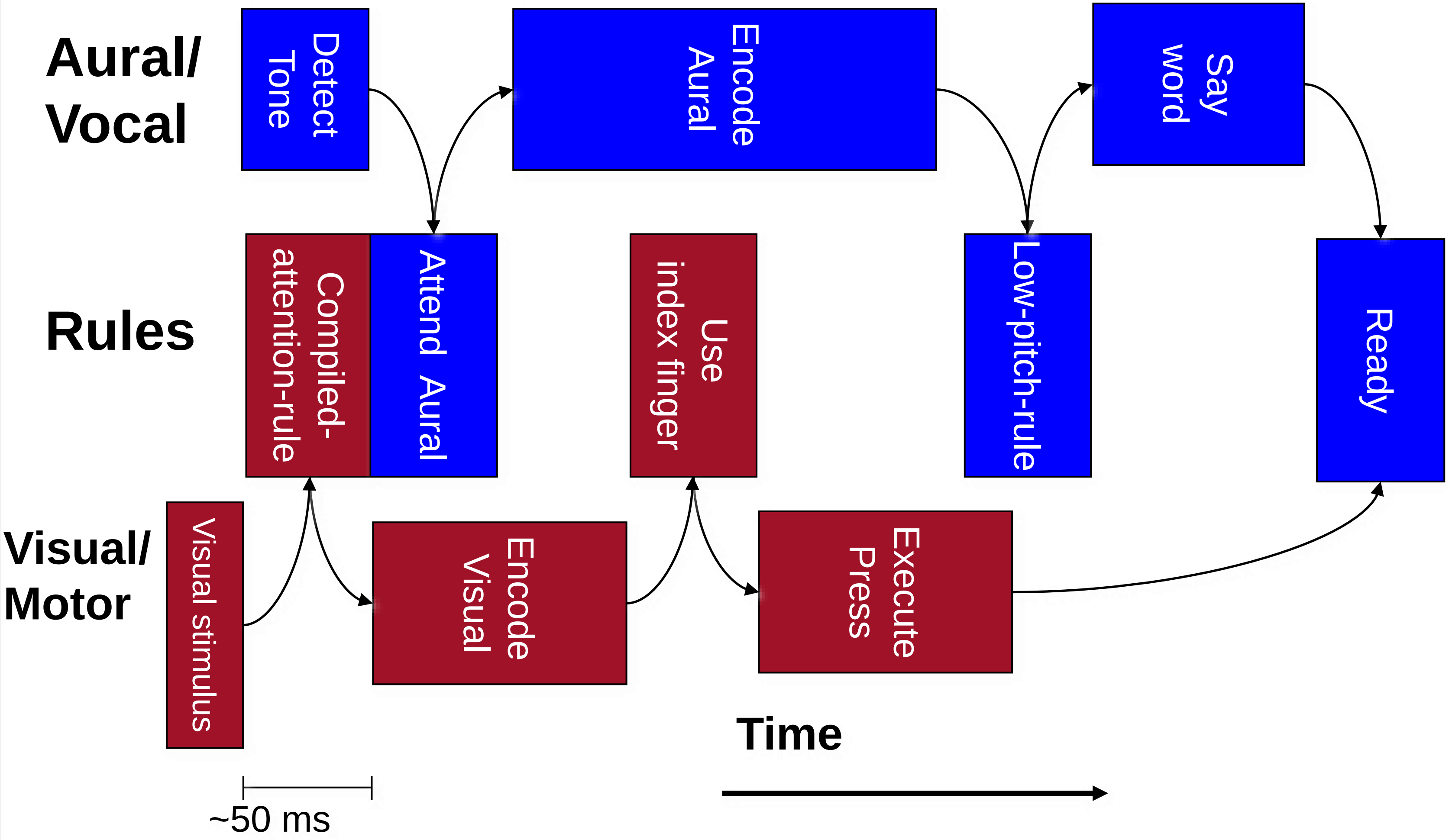
Encode Visual

Encode Visual

Execute Press

~50 ms

Time



Aural/ Vocal

Detect Tone

Encode Aural

Say word

Rules

Compiled-attention-rule

Attend Aural

Use index finger

Low-pitch-rule

Ready

Visual/ Motor

Visual stimulus

Encode Visual

Execute Press

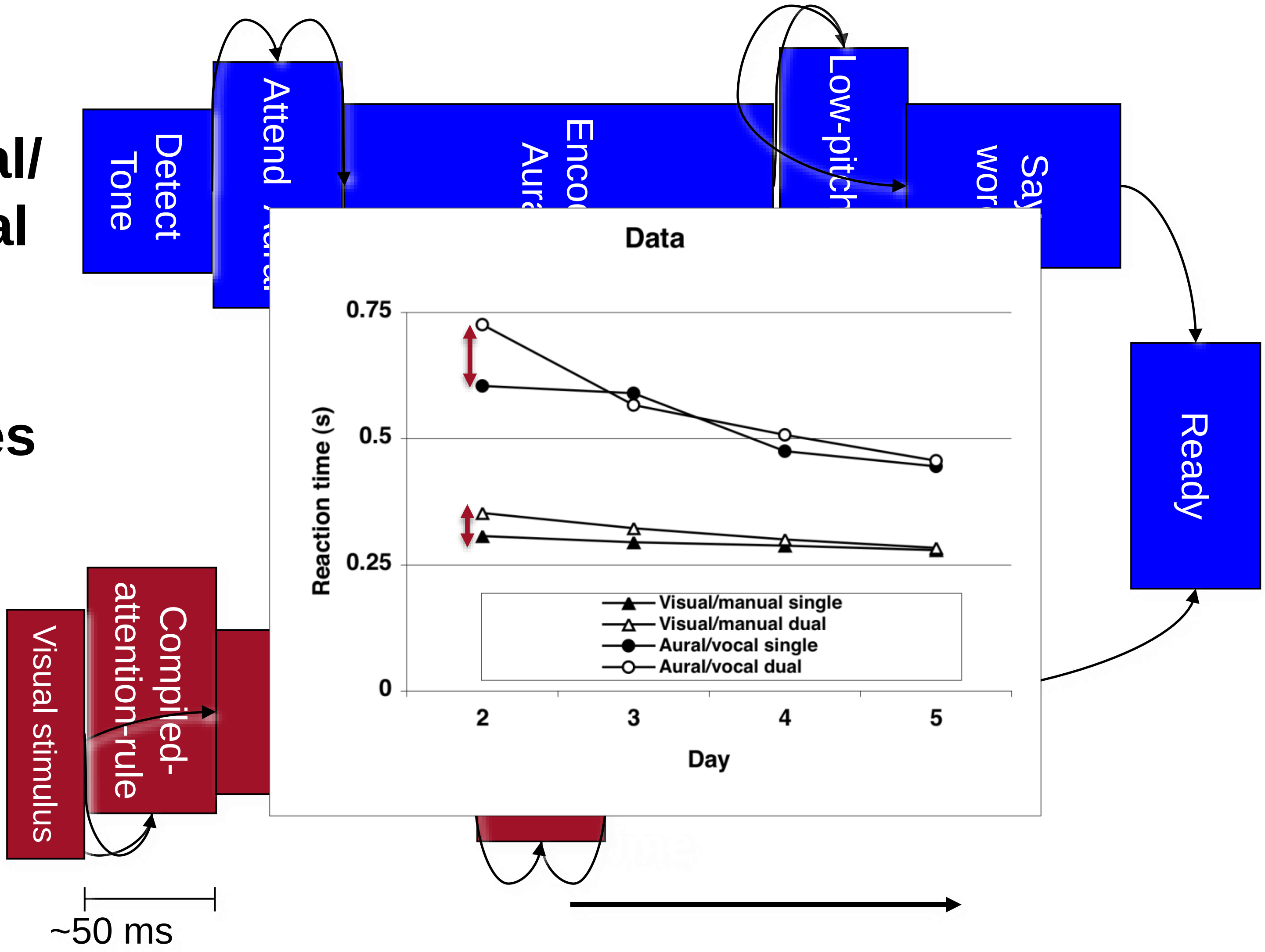
~50 ms

Time

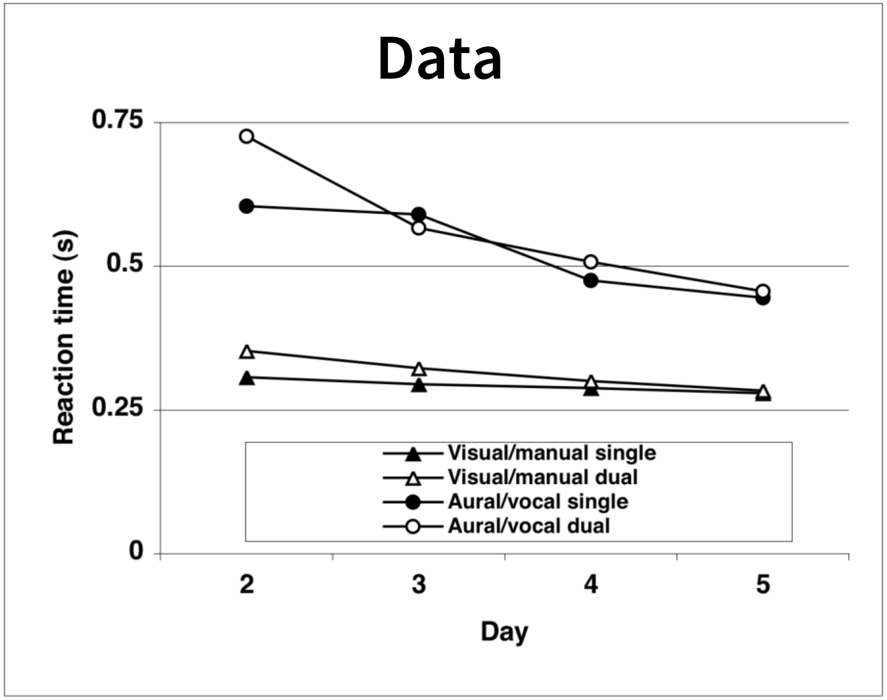
Aural/ Vocal

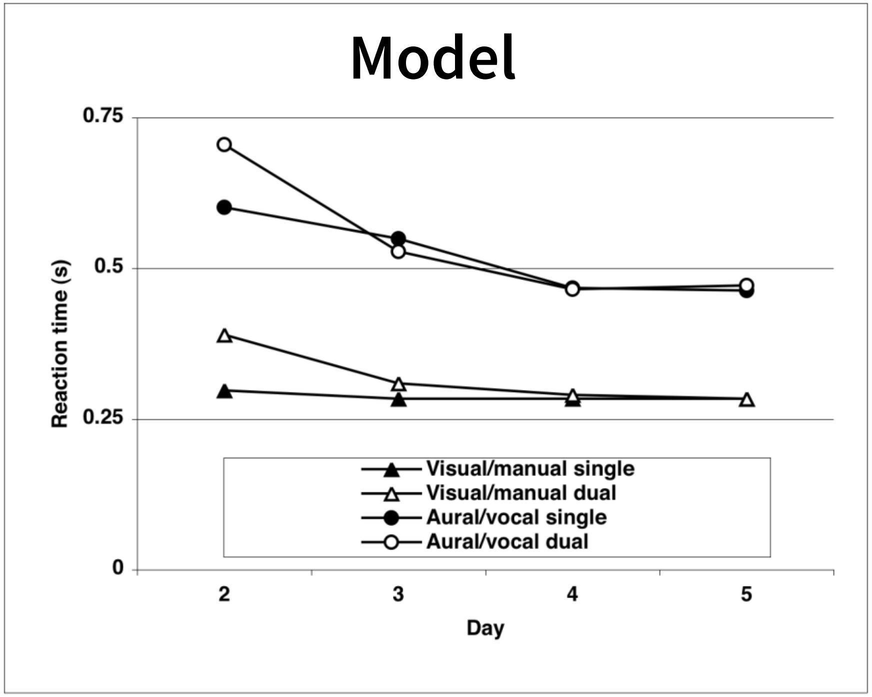
Rules

Visual/ Motor



Model/data comparison


Data


Model

(Taatgen, 2005)

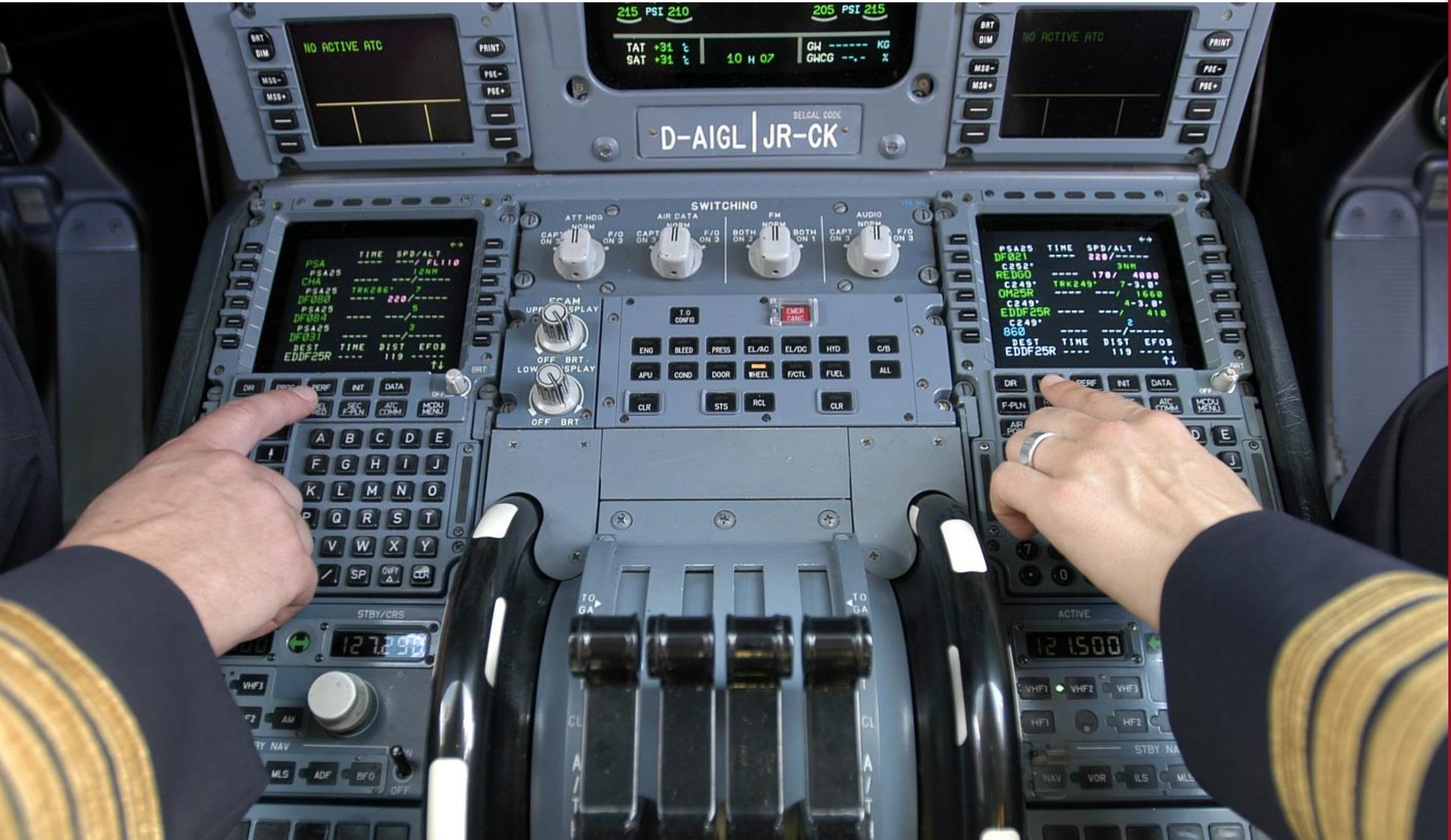
Lessons learned

- Pre-specifying top-down control in concurrent multitasking is very hard…
- and leads to brittle behavior…
- …but learning bottom-up control automatically finds a solution adhering to the minimal control principle

Today

- Learning from instructions
- Cognitive Control and the Minimal Control Principle
- Putting it all together:
  Programming the Flight Management System

Flight Management System

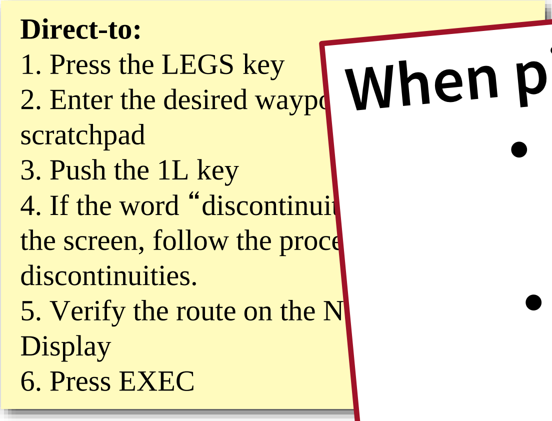


How do pilots learn the FMS?

Class Room

Simulator

Real Life



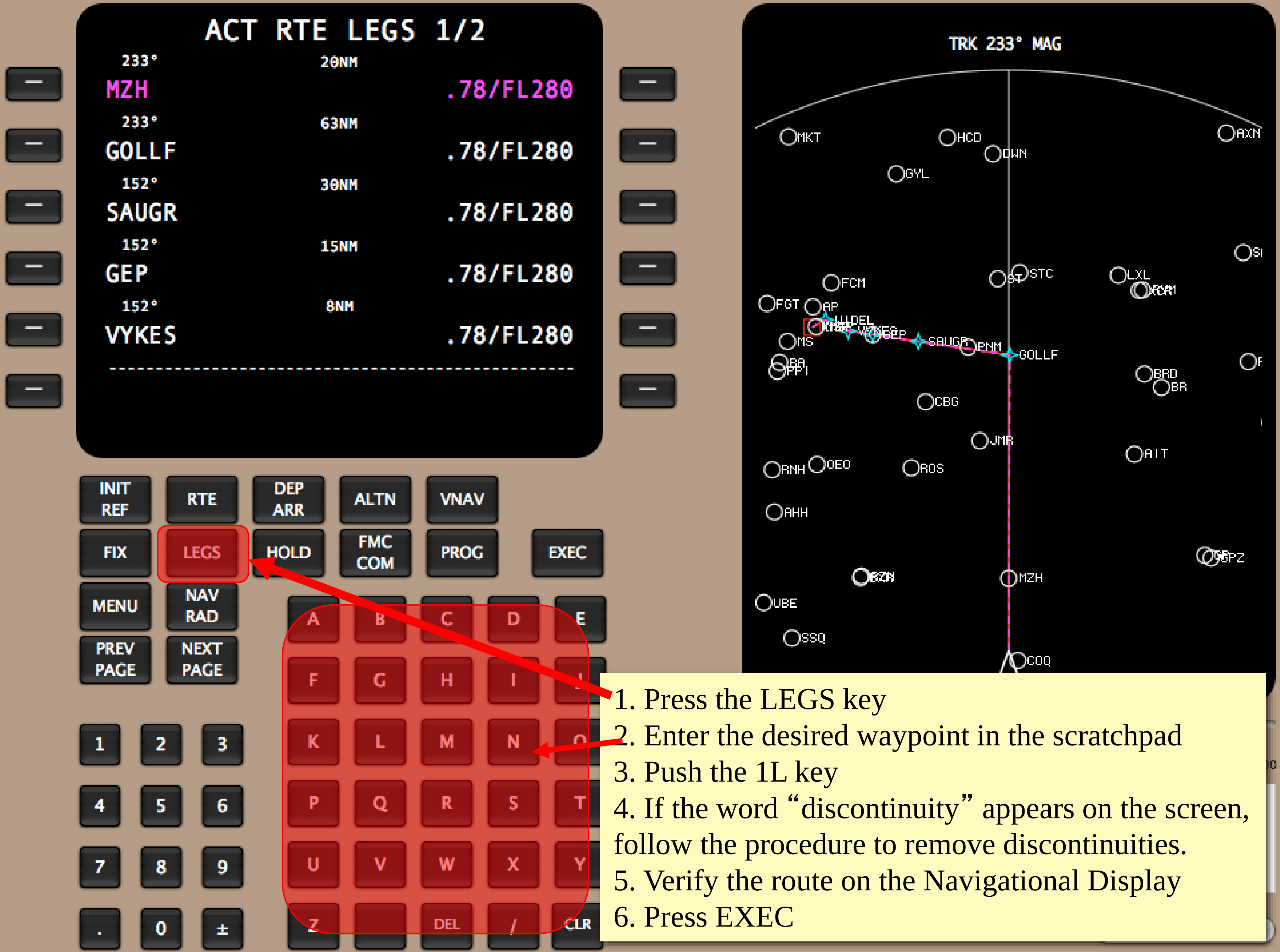
Direct-to:
1. Press the LEGS key
2. Enter the desired waypo
scratchpad
3. Push the 1L key
4. If the word "discontinui
the screen, follow the proce
discontinuities.
5. Verify the route on the N
Display
6. Press EXEC

When pilots move to the simulator they:

- forgot particular steps of a procedure
- are not able to finish a partly completed procedure
- cannot generalize

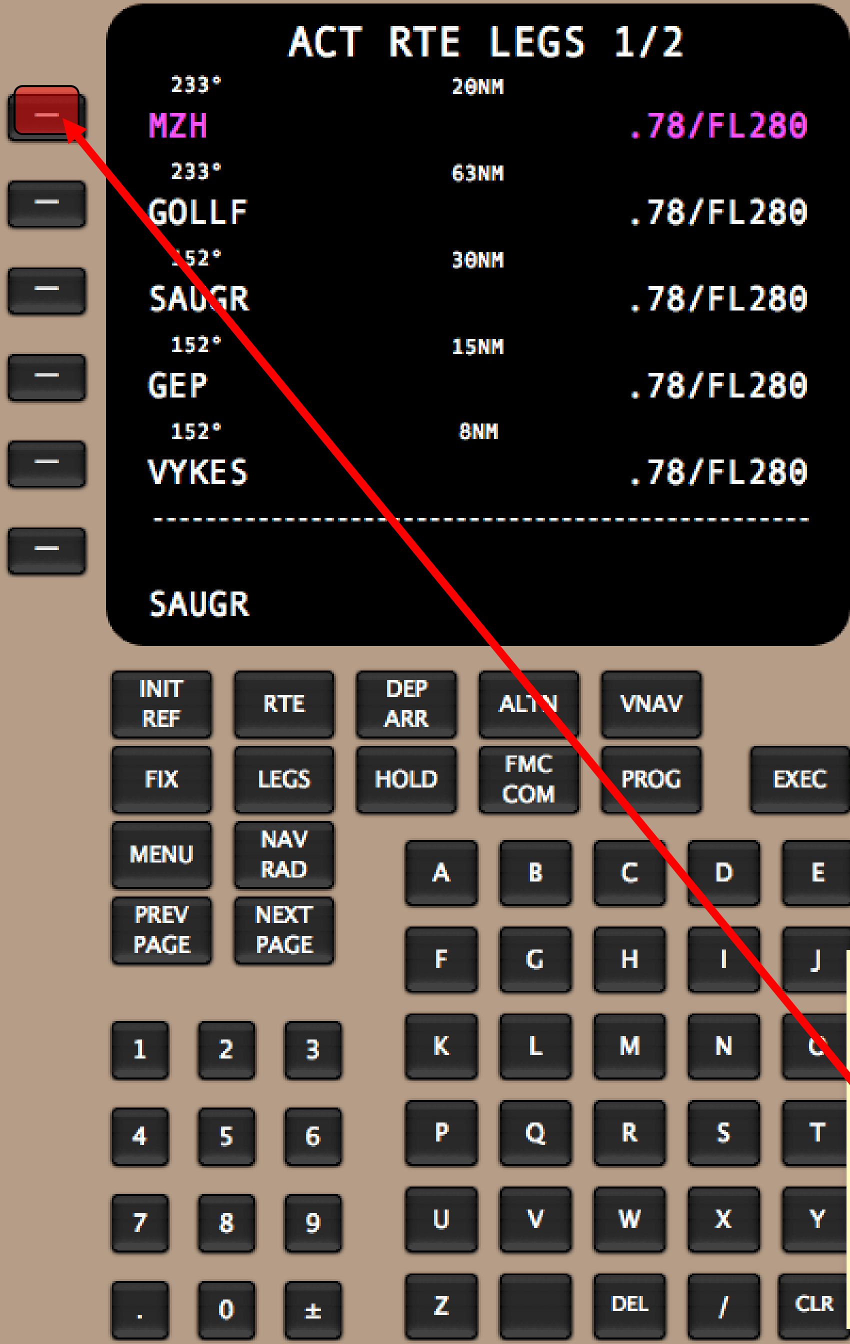
- 25 out of 102 procedures
- Pilots are assumed to infer the others

(Taatgen, Huss & Anderson, 2006)

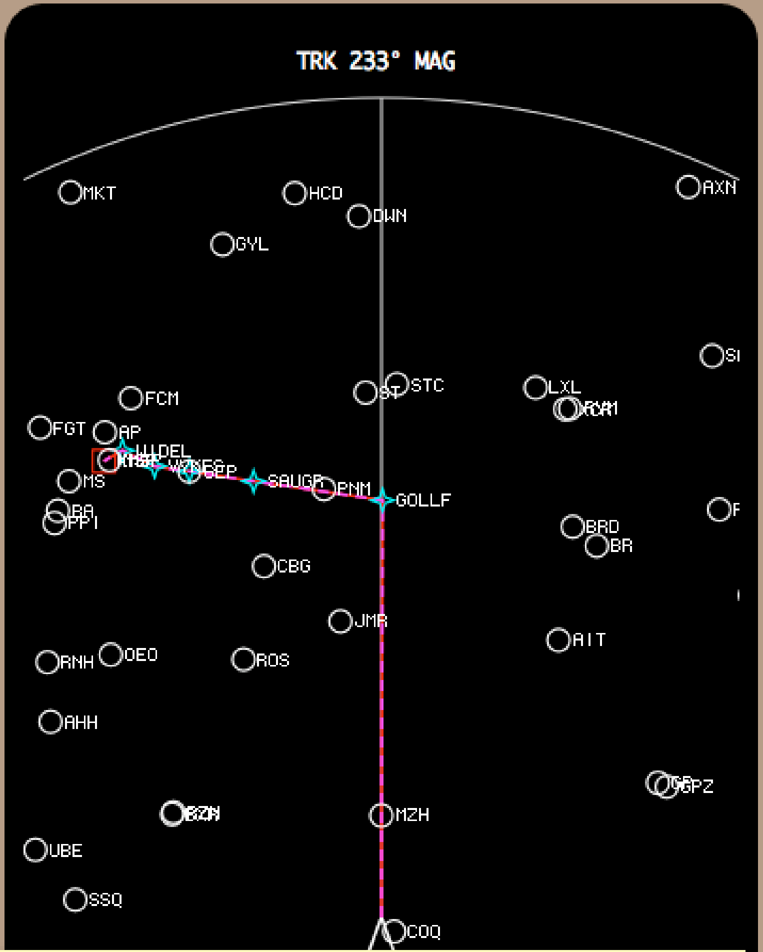


```
ACT RTE LEGS 1/2
 233°           20NM
MZH                  .78/FL280
 233°           63NM
GOLLF                .78/FL280
 152°           30NM
SAUGR                .78/FL280
 152°           15NM
GEP                  .78/FL280
 152°            8NM
VYKES                .78/FL280
--------------------------------
```

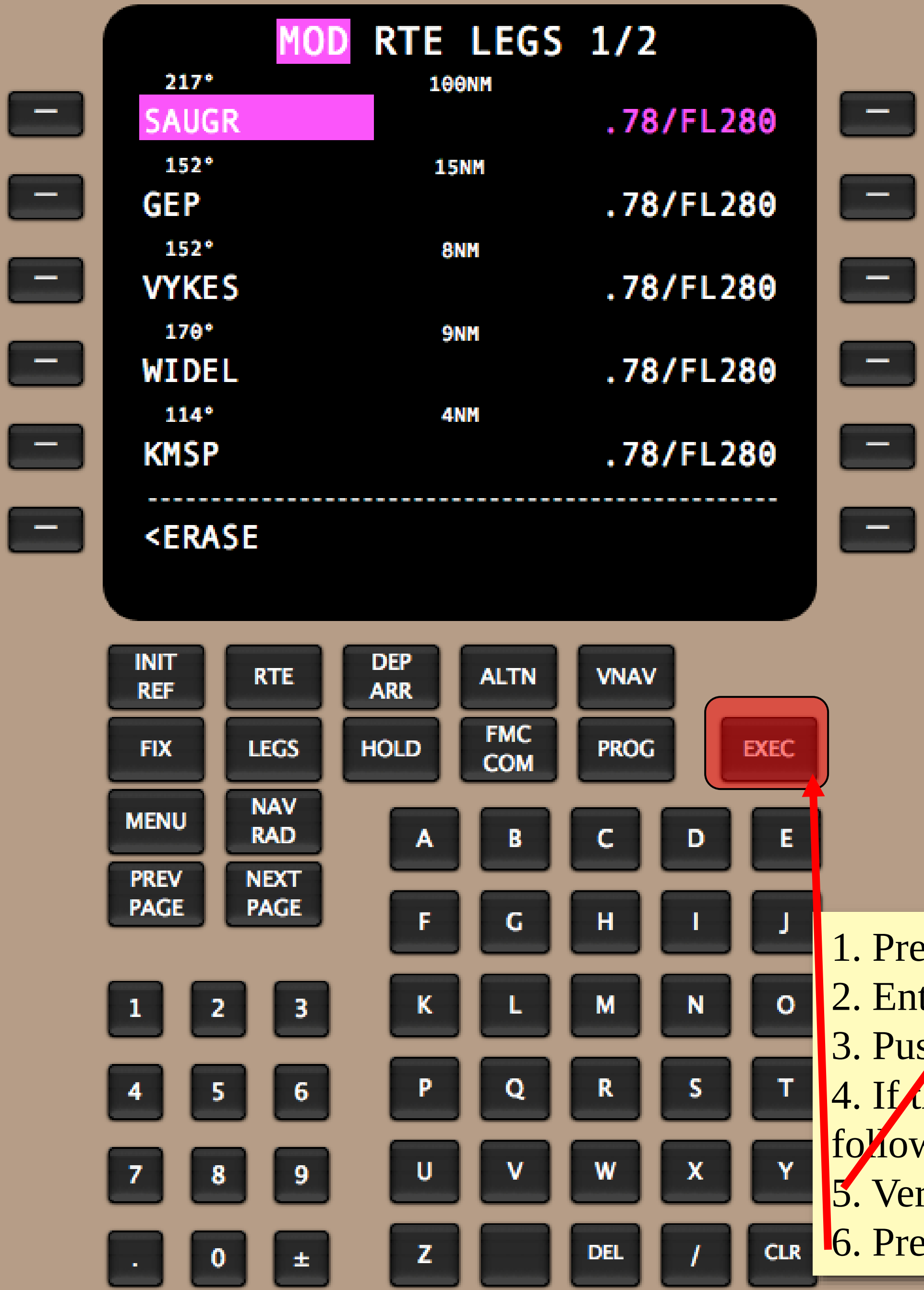
1. Press the LEGS key
2. Enter the desired waypoint in the scratchpad
3. Push the 1L key
4. If the word “discontinuity” appears on the screen, follow the procedure to remove discontinuities.
5. Verify the route on the Navigational Display
6. Press EXEC



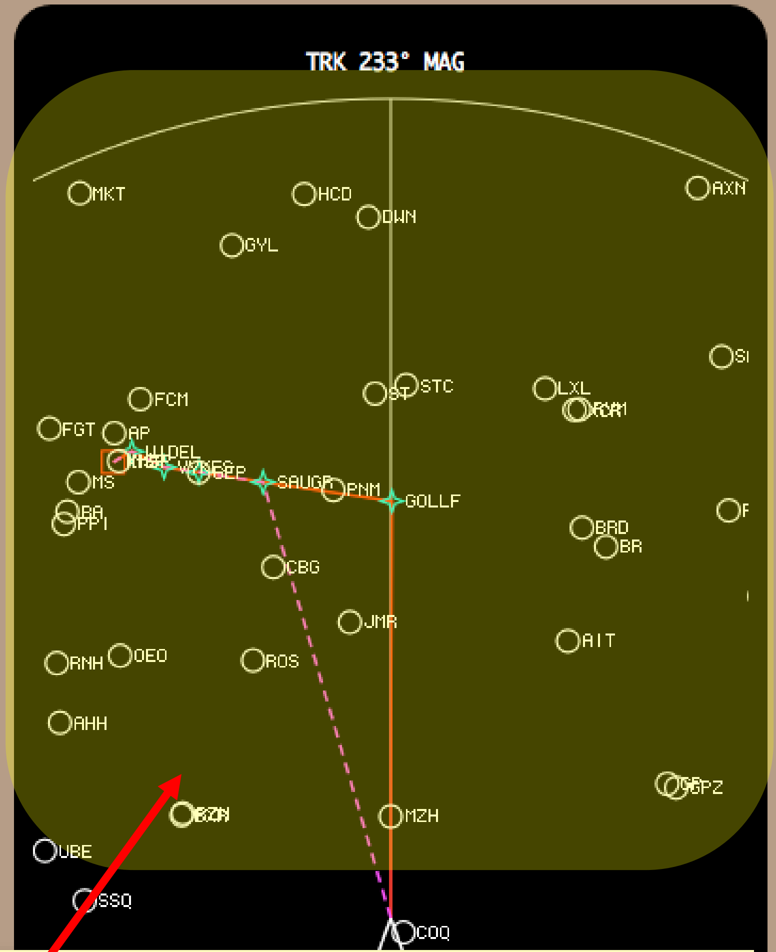
```
        ACT RTE LEGS 1/2
 233°          20NM
MZH                 .78/FL280
 233°          63NM
GOLLF               .78/FL280
 152°          30NM
SAUGR               .78/FL280
 152°          15NM
GEP                 .78/FL280
 152°           8NM
VYKES               .78/FL280
------------------------------

SAUGR
```



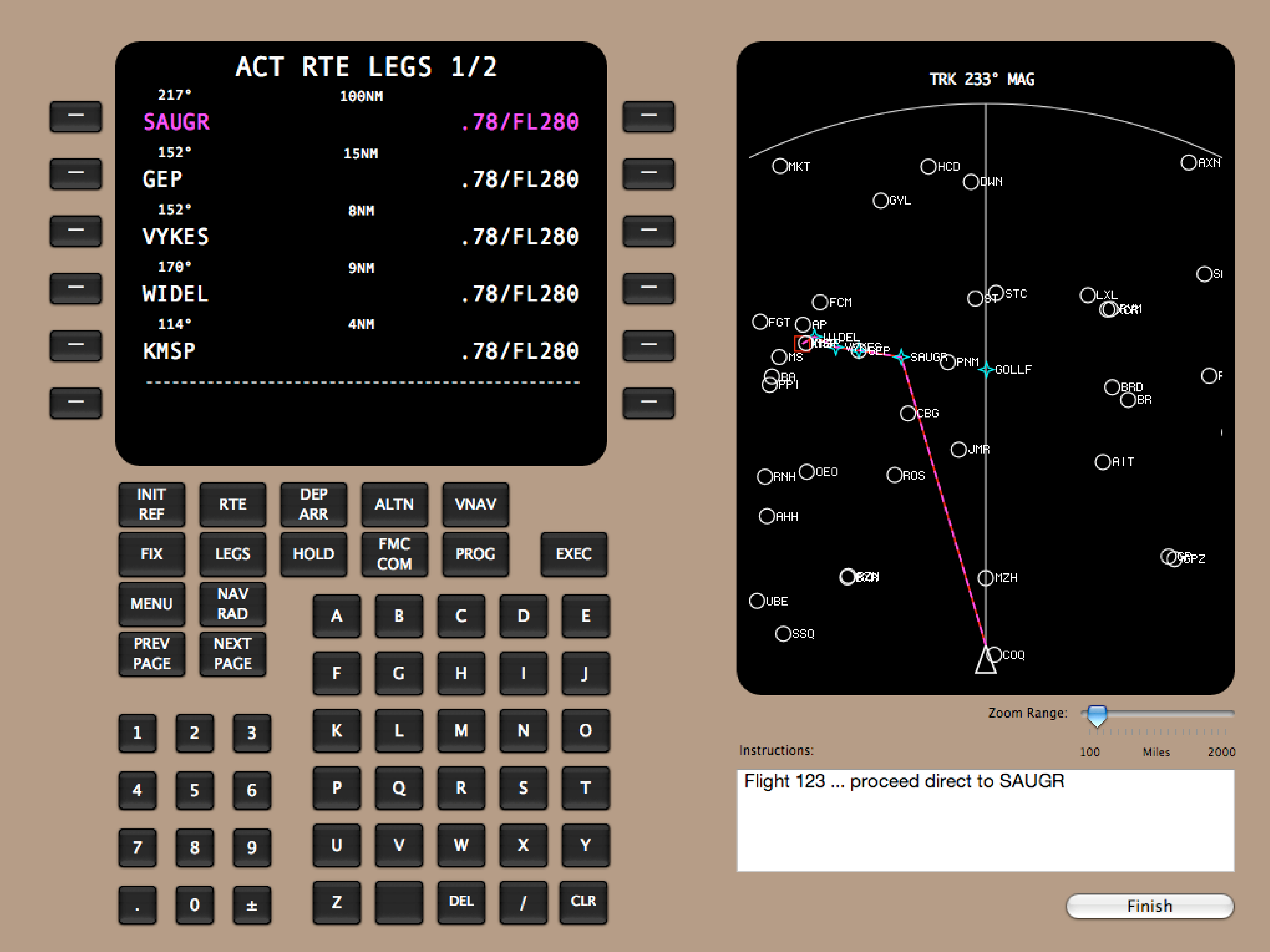
1. Press the LEGS key
2. Enter the desired waypoint in the scratchpad
3. Push the 1L key
4. If the word “discontinuity” appears on the screen, follow the procedure to remove discontinuities.
5. Verify the route on the Navigational Display
6. Press EXEC



```
     MOD RTE LEGS 1/2
 217°          100NM
SAUGR                .78/FL280
 152°          15NM
GEP                  .78/FL280
 152°          8NM
VYKES                .78/FL280
 170°          9NM
WIDEL                .78/FL280
 114°          4NM
KMSP                 .78/FL280
------------------------------
<ERASE
```



1. Press the LEGS key
2. Enter the desired waypoint in the scratchpad
3. Push the 1L key
4. If the word “discontinuity” appears on the screen, follow the procedure to remove discontinuities.
5. Verify the route on the Navigational Display
6. Press EXEC

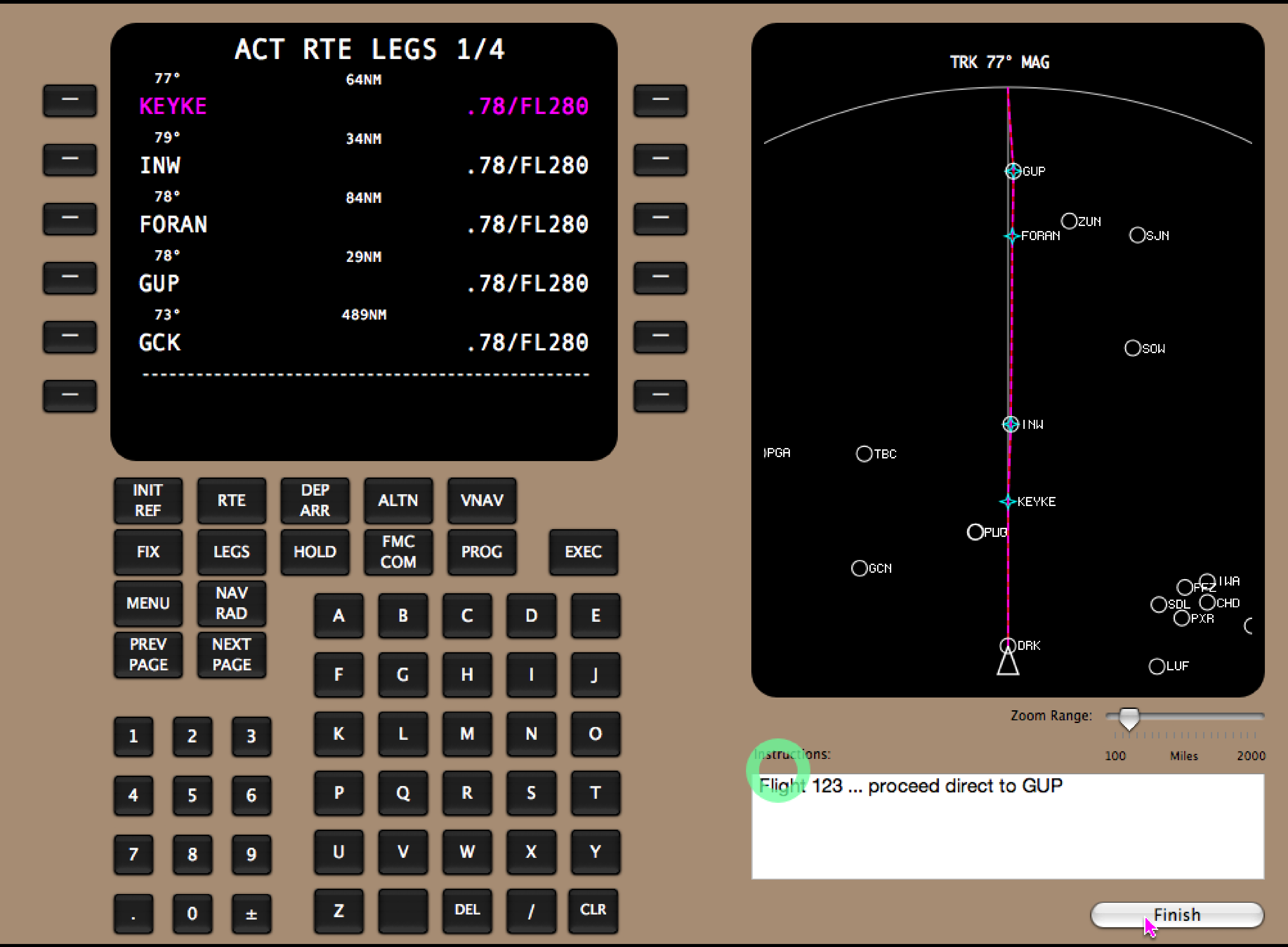


ACT RTE LEGS 1/2

```
 217°        100NM
SAUGR               .78/FL280
 152°         15NM
GEP                 .78/FL280
 152°          8NM
VYKES               .78/FL280
 170°          9NM
WIDEL               .78/FL280
 114°          4NM
KMSP                .78/FL280
------------------------------
```

Instructions:

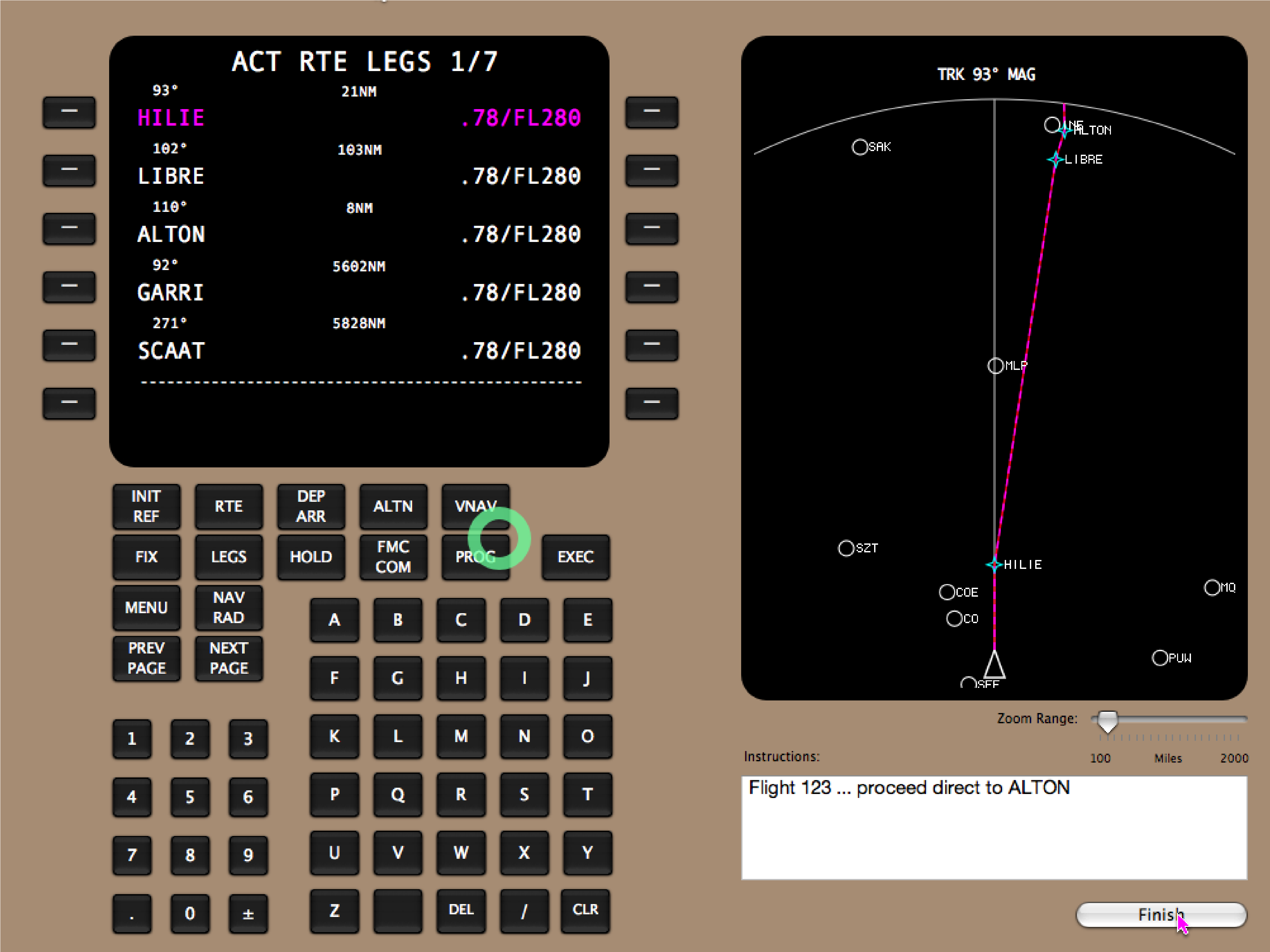
Flight 123 ... proceed direct to SAUGR



```
      ACT RTE LEGS 1/4
 77°         64NM
KEYKE               .78/FL280
 79°         34NM
INW                 .78/FL280
 78°         84NM
FORAN               .78/FL280
 78°         29NM
GUP                 .78/FL280
 73°        489NM
GCK                 .78/FL280
 ----------------------------------------
```

Instructions:

Flight 123 ... proceed direct to GUP



```
          ACT RTE LEGS 1/7
 93°            21NM
HILIE                   .78/FL280
 102°           103NM
LIBRE                   .78/FL280
 110°           8NM
ALTON                   .78/FL280
 92°            5602NM
GARRI                   .78/FL280
 271°           5828NM
SCAAT                   .78/FL280
---------------------------------
```

TRK 93° MAG

Zoom Range:

100 Miles 2000

Instructions:

Flight 123 ... proceed direct to ALTON

Experiment with instructional manipulation

List

1. Press the LEGS key
2. Enter the desired waypoint in the scratchpad
3. Push the 1L key
4. If the word “discontinuity” appears on the screen, follow the procedure to remove discontinuities.
5. Verify the route on the Navigational Display
6. Press EXEC

UNITED

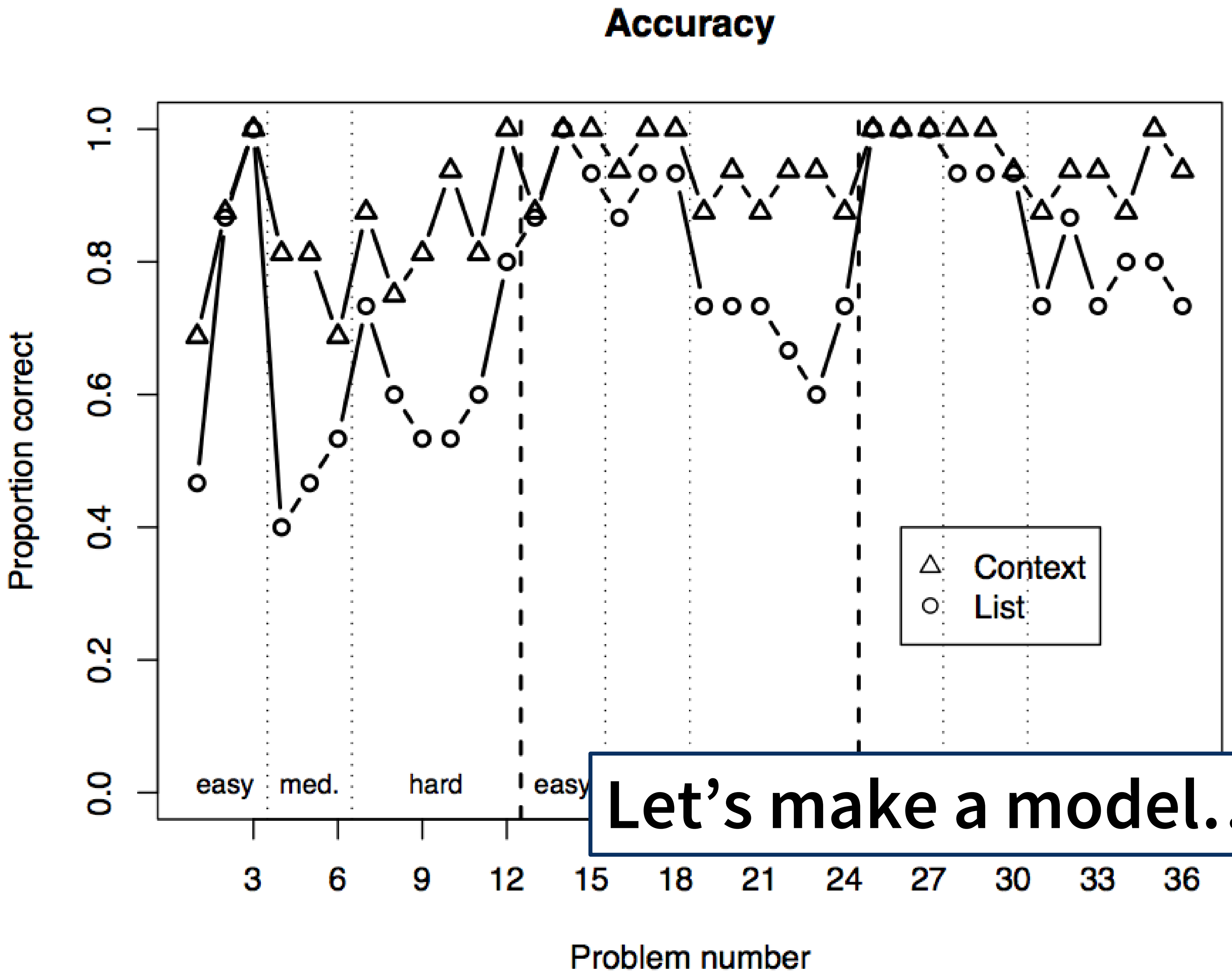
Context

- If you want to change the route and you are not yet on the LEGS page, then press the LEGS key in order to go to the LEGS page.
- If you want to modify a waypoint, you enter the waypoint to replace it with into the scratchpad, and then press the line key corresponding to the waypoint you want to modify

Hypotheses

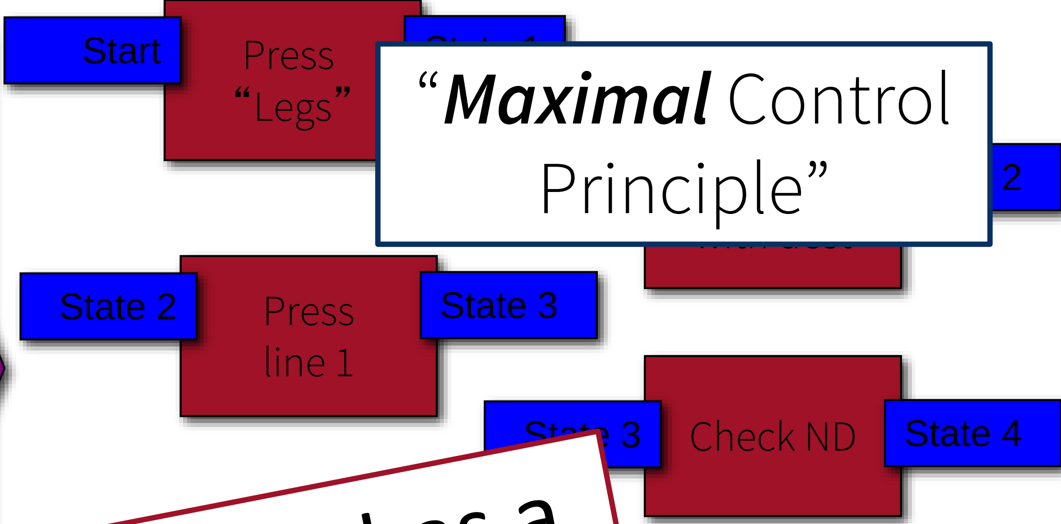
- People will do better with the context instructions, because they lead to a strategy that requires less internal control
- This will be especially true for more complex problems in which generalization is needed

- Structure of the experiment is 3 blocks of:
  - 3 Easy problems using 1 procedure
  - 3 Medium problems using 2 procedures
  - 6 Hard problems that require some generalization



Let’s make a model…

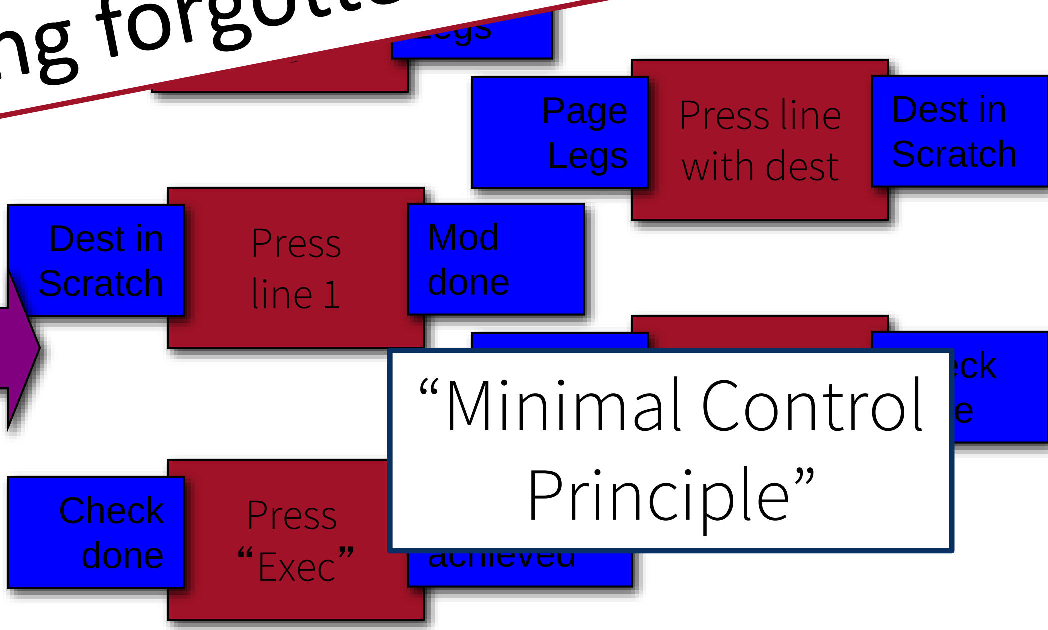
1. Press the LEGS key
2. Enter the desired waypoint in the scratchpad
3. Push the 1L key
4. If the word “discontinuity” appears on the screen,
follow the procedure to remove discontinuities.
5. Verify the route on the Navigational Display
6. Press EXEC



“***Maximal*** Control Principle”

Each instruction has a 25% probability of being forgotten!

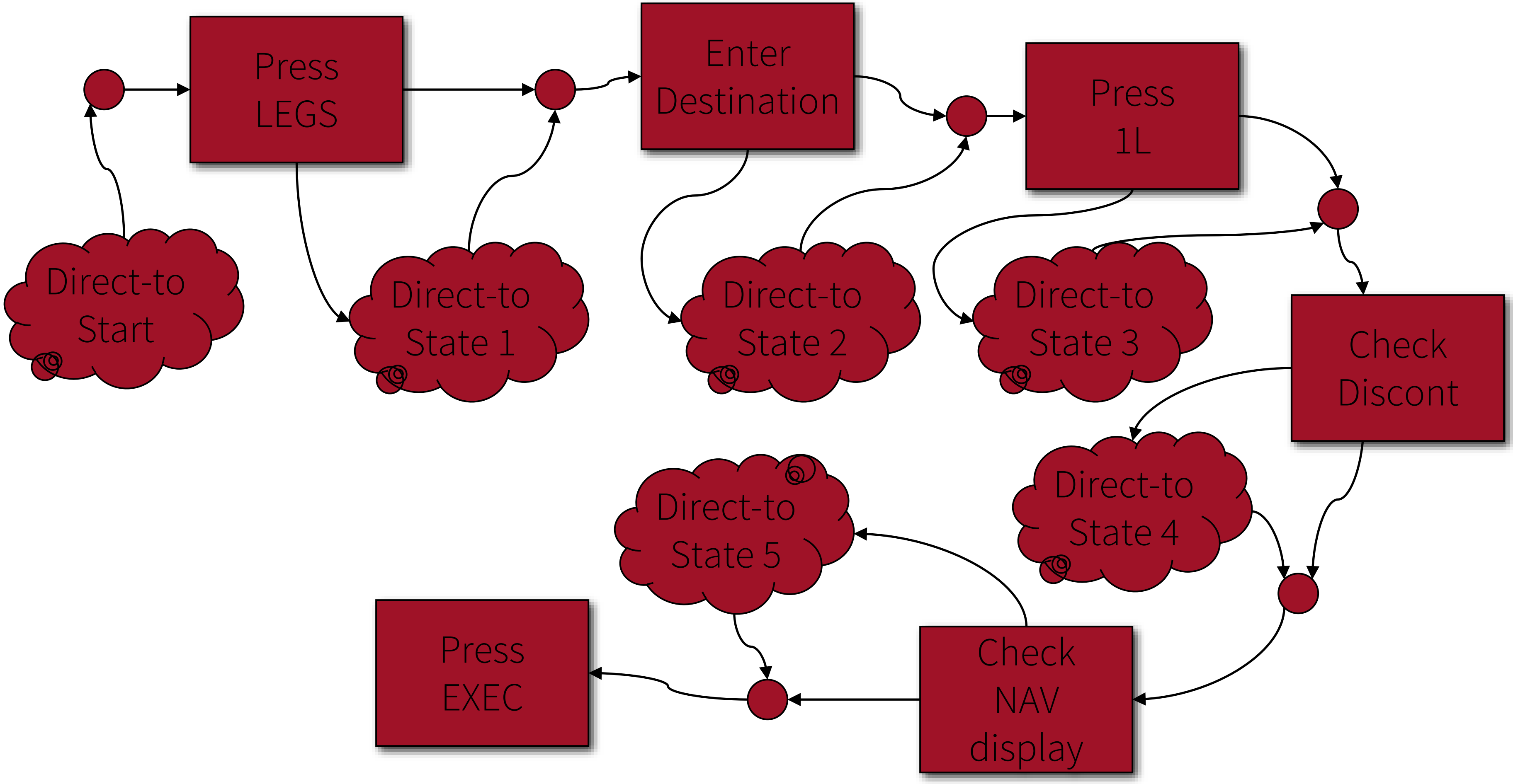
- If you want to change the r you are not yet on the LEGS page, then press the LEGS key in order to go to the LEGS page.
- If you want to modify a waypoint, you enter the waypoint to replace it with into the scratchpad, and then press the line key corresponding to the waypoint you want to modify



“Minimal Control Principle”

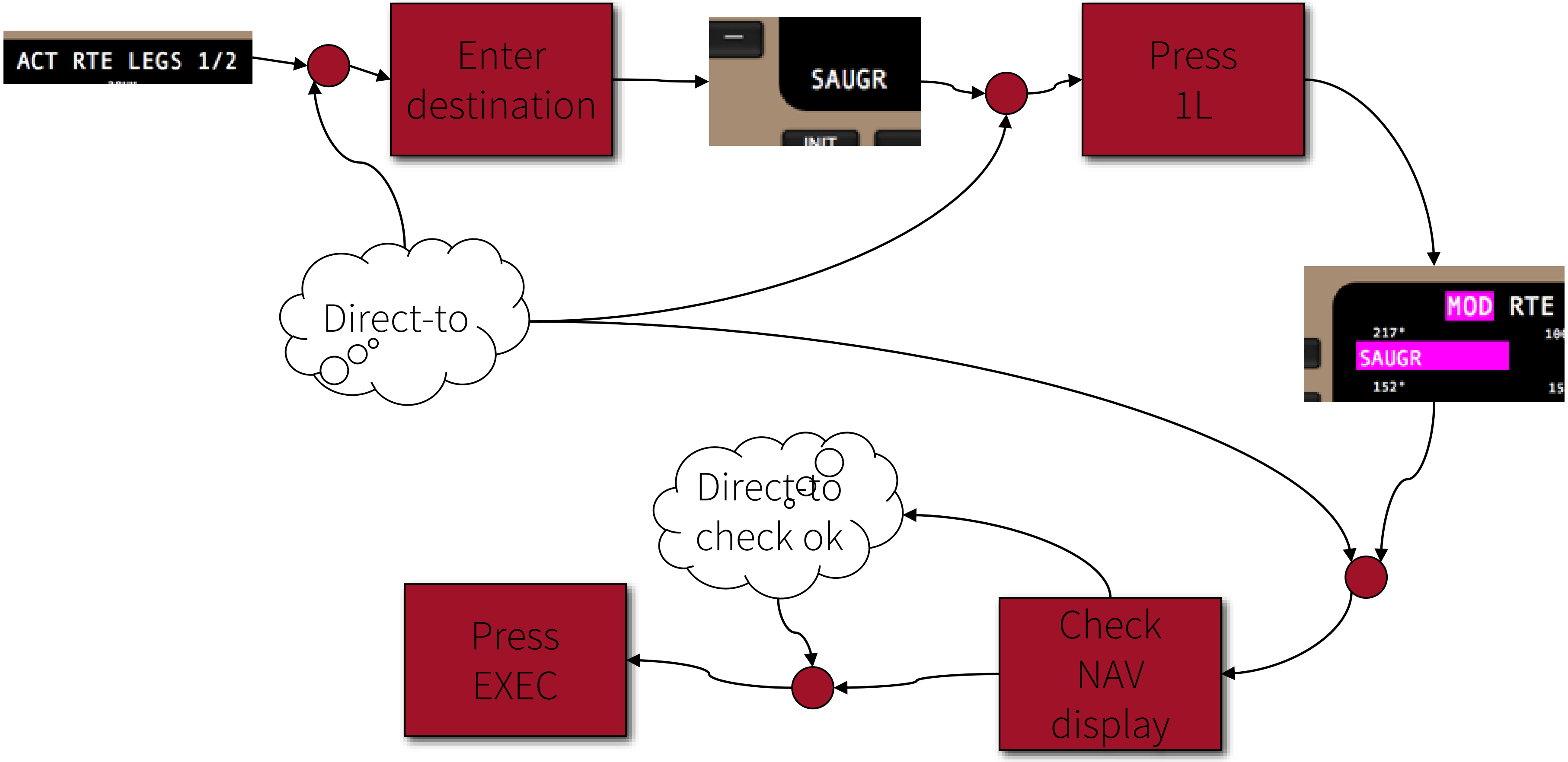
List instructions lead to top-down control

"***Maximal*** Control Principle"



Context instructions lead to bottom-up control

"Minimal Control Principle"



Enter destination

Press 1L

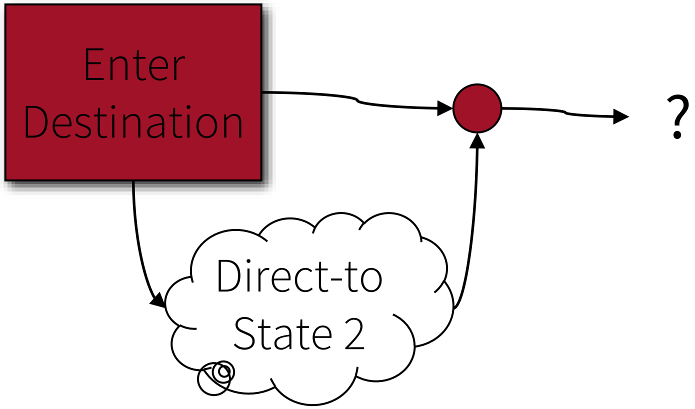
Direct-to

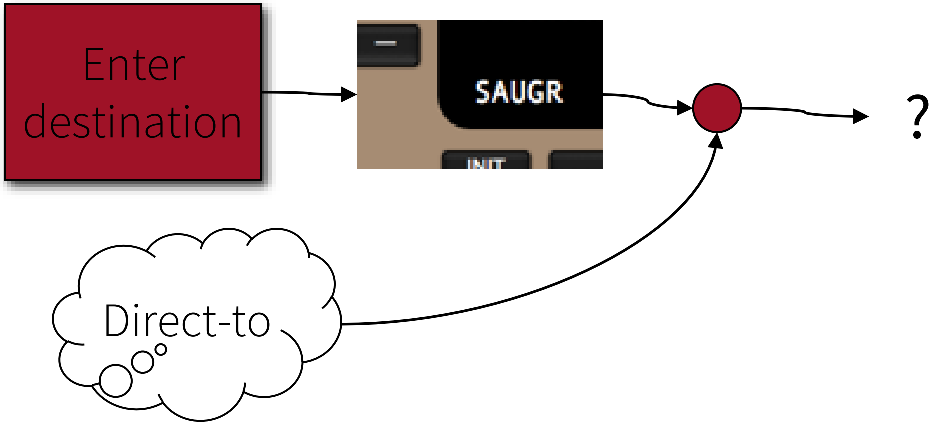
Direct-to check ok

Check NAV display

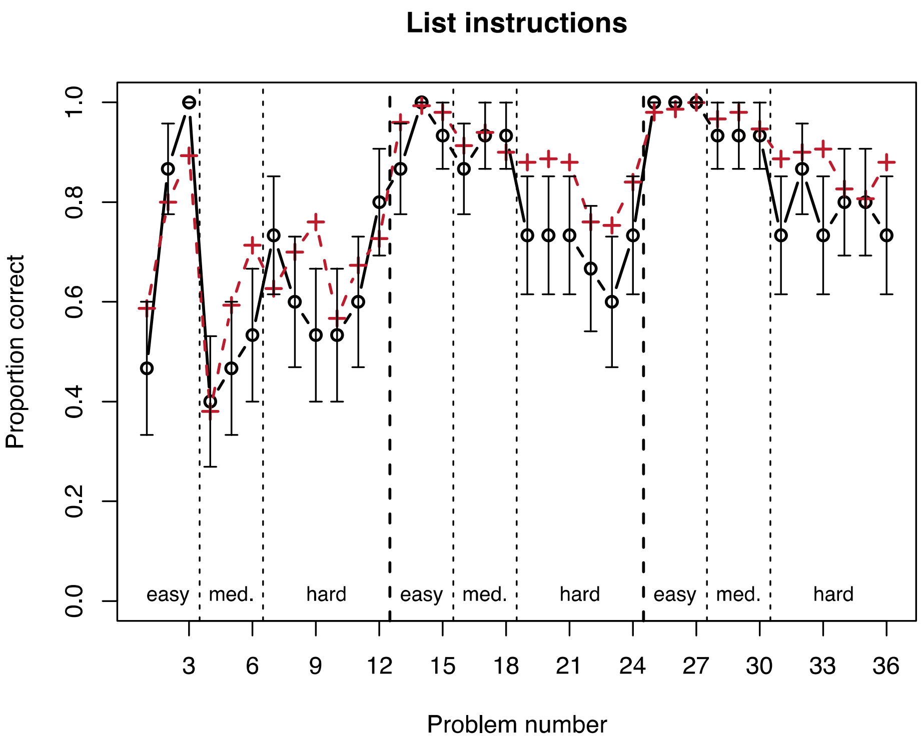
Press EXEC

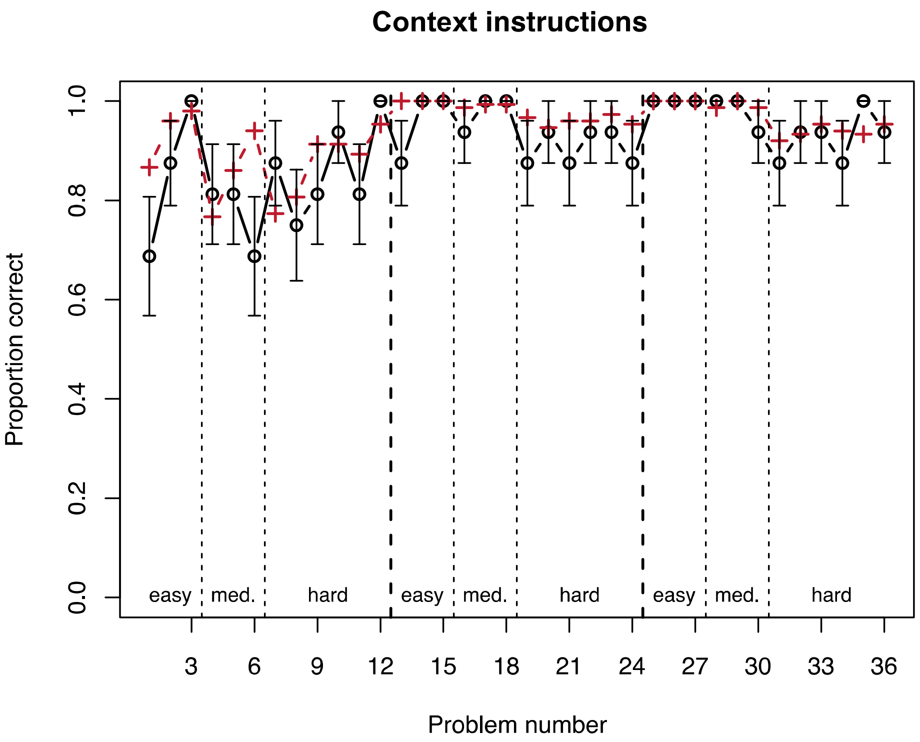
Missing operators: Trial-and-Error





Accuracy

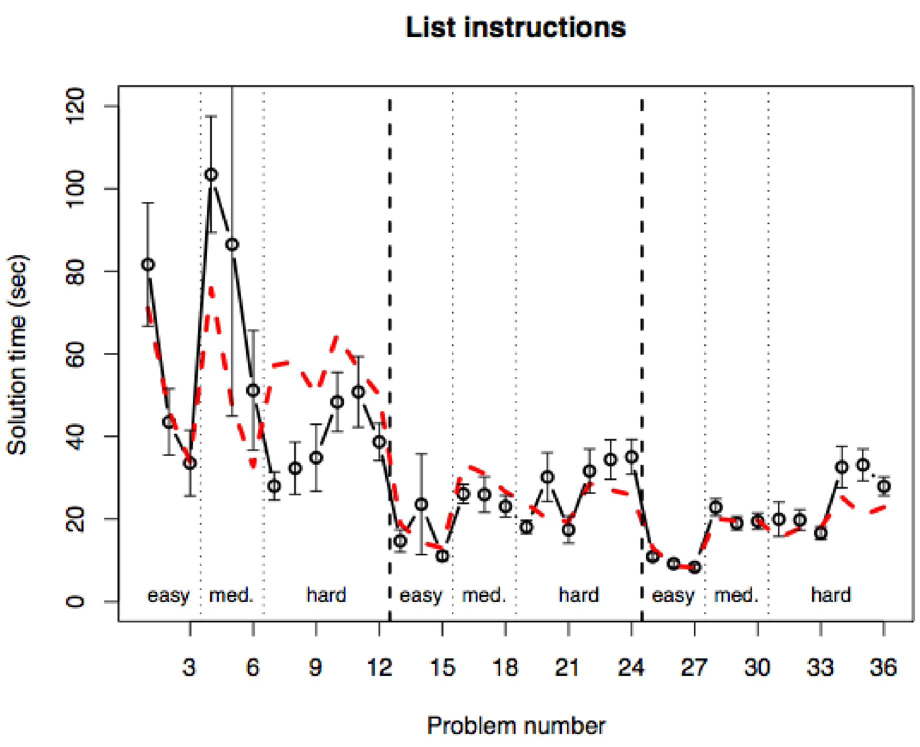

List instructions

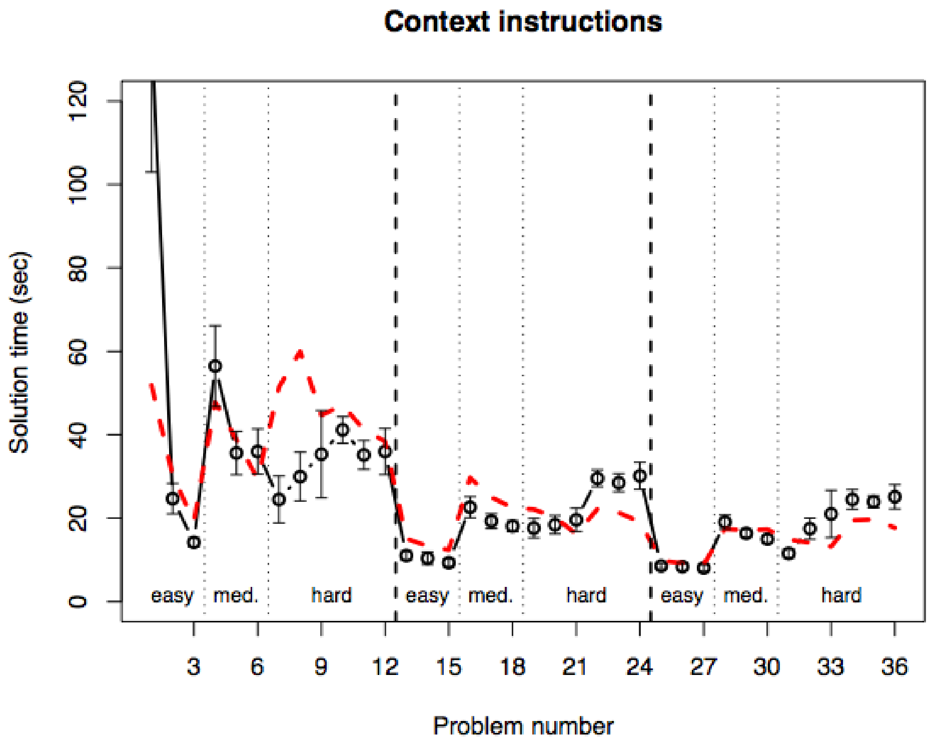

Context instructions

data

model

Solution times

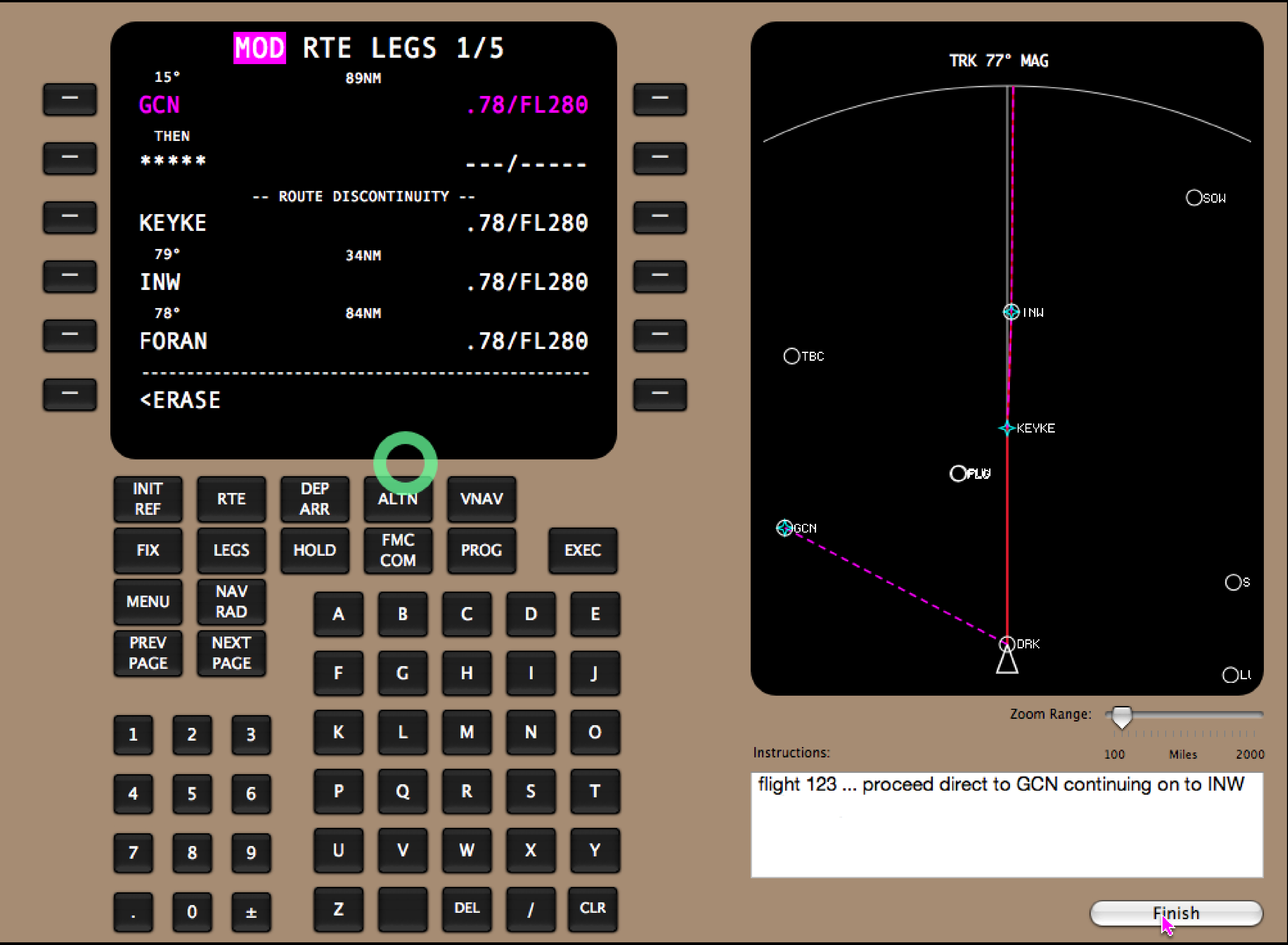




Second experiment

- Same instructions
- New problems in which the procedure was partially completed, *sometimes with an error*

- Model was unchanged because the instructions were the same --> Prediction!

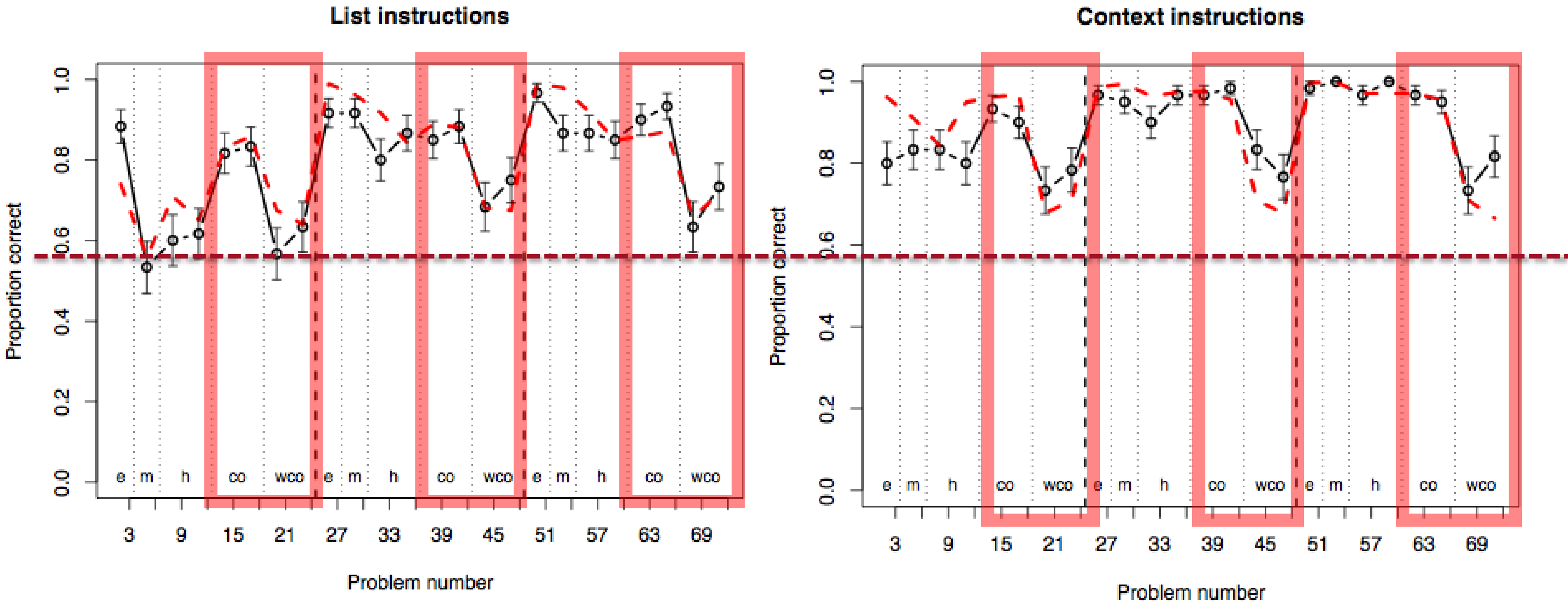


```
         MOD RTE LEGS 1/5
 15°            89NM
GCN                     .78/FL280
 THEN
*****                   ---/-----
    -- ROUTE DISCONTINUITY --
KEYKE                   .78/FL280
 79°            34NM
INW                     .78/FL280
 78°            84NM
FORAN                   .78/FL280
------------------------------------
<ERASE
```

Instructions:

flight 123 ... proceed direct to GCN continuing on to INW

Results



- The *same* model was applied to a second experiment, providing accurate *predictions*
- Instructions that support minimal control lead to better performance: relevant for design of instructions and interfaces

Conclusions

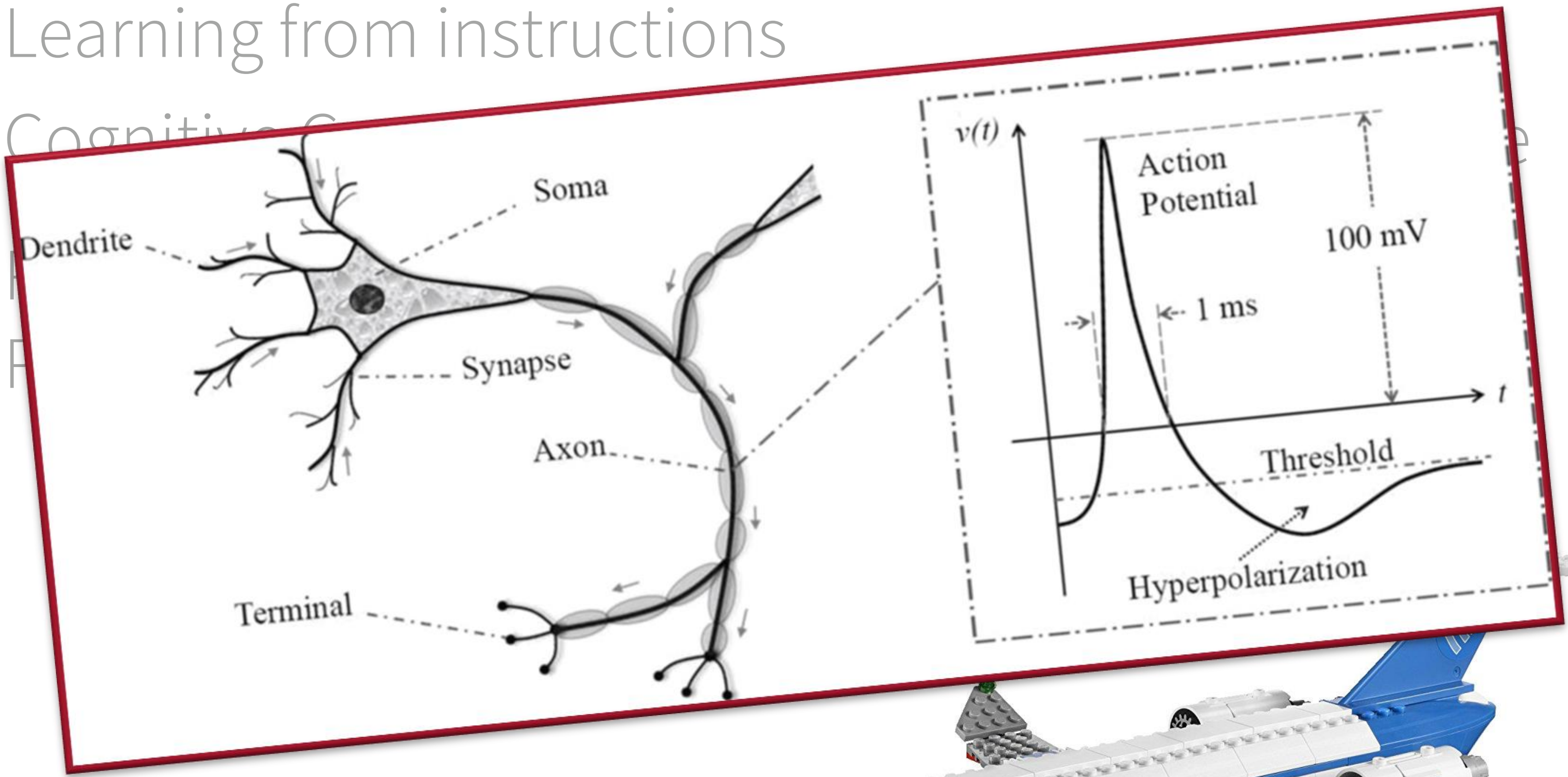
- Cognitive Control in complex tasks is a combination of internal states and the environment
- This depends on task instructions:
  - Context instructions improved behavior on the FMS
- We can model how control is learned

Today

- Learning from instructions
- Cognitive Control and the Minimal Control Principle
- Putting it all together:
  Programming the Flight Management System

Next week:

- Learning from instructions
- Cognitive C



Have a good weekend!

university of groningen