Cognitive Models of Programming

Let’s wrap up our Tuesday activity!

What we’re thinking about today

- How can we design studies with programmers to get information about their mental models?

COGNITIVE PSYCHOLOGY 19, 295–341 (1987)

Stimulus Structures and Mental Representations in Expert Comprehension of Computer Programs

NANCY PENNINGTON

Graduate School of Business, University of Chicago

Comprehension of computer programs involves detecting or inferring different kinds of relations between program parts. Different kinds of programming knowledge facilitate detection and representation of the different textual relations. The present research investigates the role of programming knowledge in program comprehension and the nature of mental representations of programs; specifically, whether procedural (control flow) or functional (goal hierarchy) relations dominate programmers' mental representations of programs. In the first study, 80 professional programmers were tested on comprehension and recognition of short computer program texts. The results suggest that procedural rather than functional units form the basis of expert programmers' mental representations, supporting work in other areas of text comprehension showing the importance of text structure knowledge in understanding. In a second study 40 professional programmers studied and modified programs of moderate length. Results support conclusions from the first study that programs are first understood in terms of their procedural episodes. However, results also suggest that a programmer's task goals may influence the relations that dominate mental representations later in comprehension.

Computer programming is a complex cognitive task composed of a va-

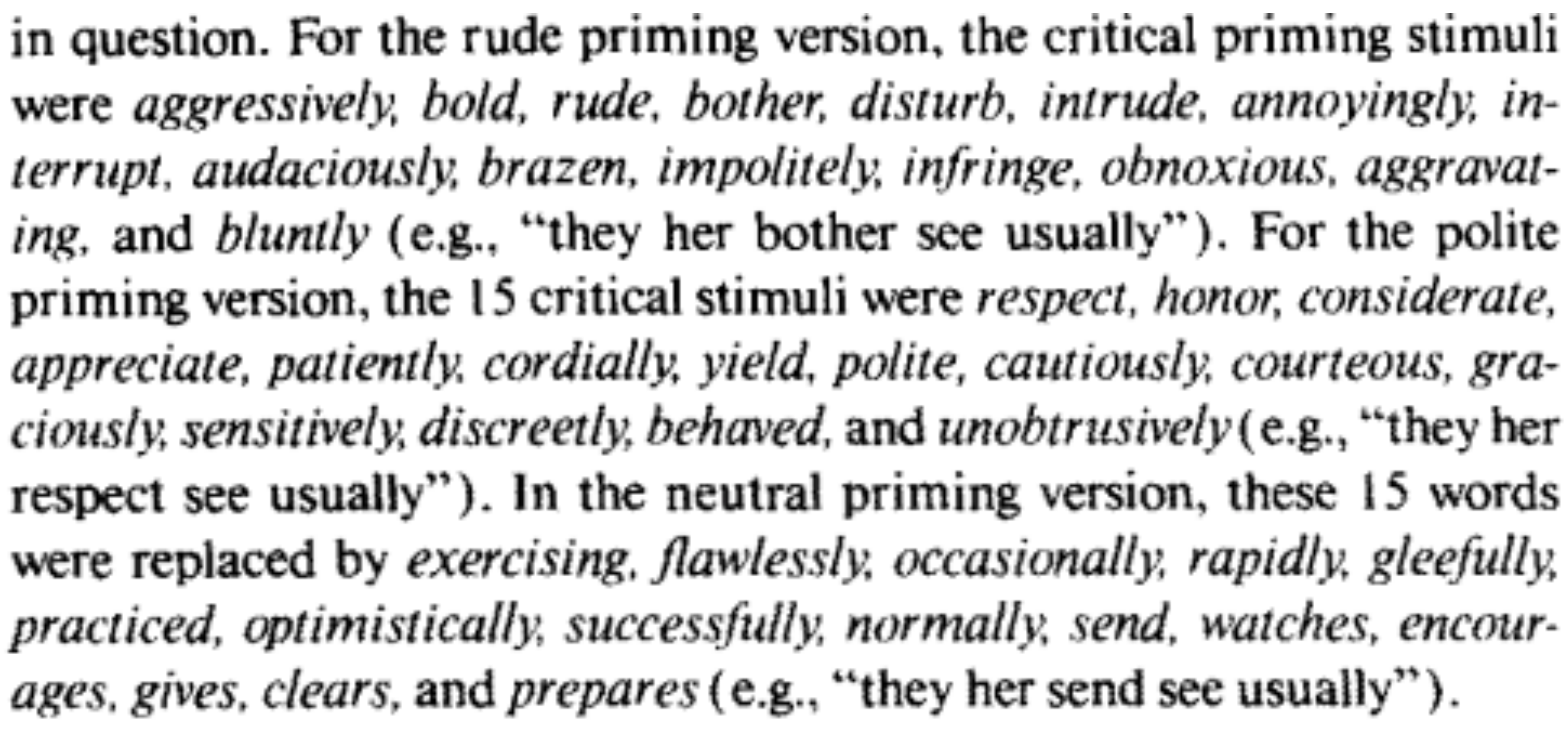
Animating question: What mental model do experts build of programs they're trying to understand?

How can we figure out what mental model the programmer has built of a program? What could give us insights into how it's structured?

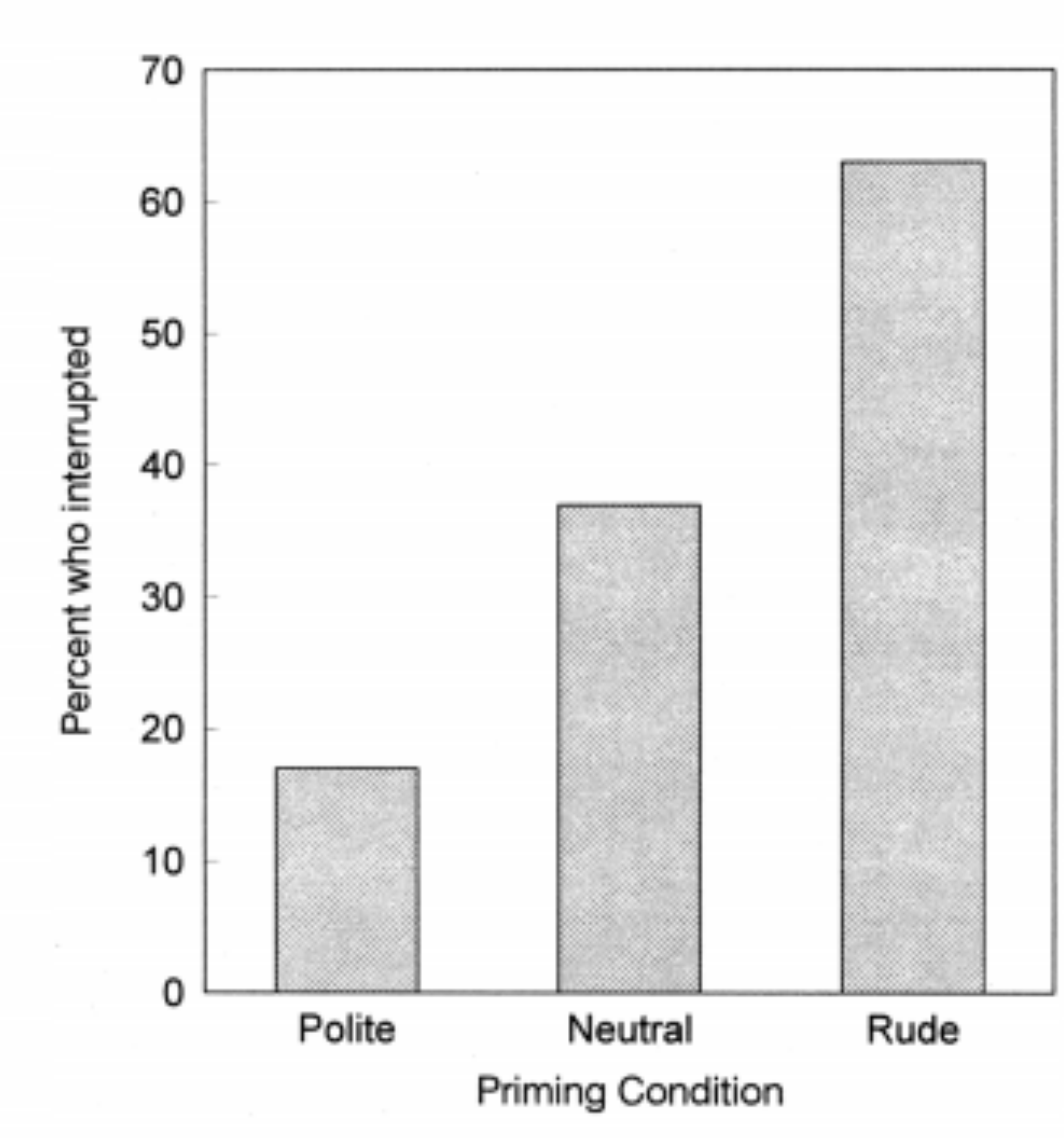
Psych 101: Priming

Example

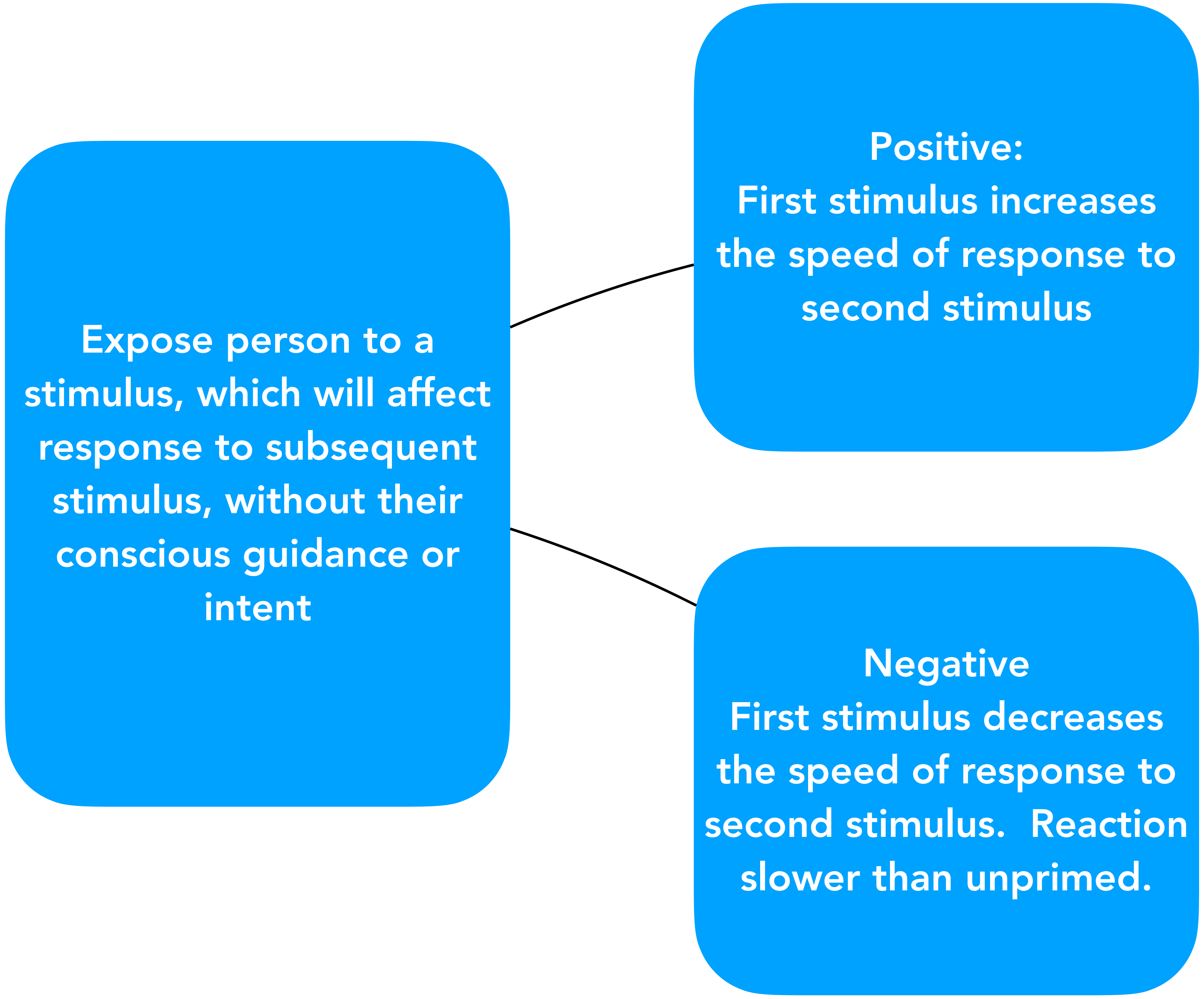
- Participants given a ‘test of language ability,’ a sentence scramble test
  - Three conditions, included words related to politeness, neutral words, or words related to rudeness



in question. For the rude priming version, the critical priming stimuli were *aggressively, bold, rude, bother, disturb, intrude, annoyingly, interrupt, audaciously, brazen, impolitely, infringe, obnoxious, aggravating*, and *bluntly* (e.g., “they her bother see usually”). For the polite priming version, the 15 critical stimuli were *respect, honor, considerate, appreciate, patiently, cordially, yield, polite, cautiously, courteous, graciously, sensitively, discreetly, behaved*, and *unobtrusively* (e.g., “they her respect see usually”). In the neutral priming version, these 15 words were replaced by *exercising, flawlessly, occasionally, rapidly, gleefully, practiced, optimistically, successfully, normally, send, watches, encourages, gives, clears*, and *prepares* (e.g., “they her send see usually”).



- Be somewhat wary of ‘social priming’ or ‘automatic behavior priming’ results! Many turn out not to replicate
    - (I chose the one on the prior slide specifically because it has been replicated. And because I think it’s cool.)
- But we’re mostly interested here in positive and negative priming, which is about when priming affects the speed of processing.



Expose person to a stimulus, which will affect response to subsequent stimulus, without their conscious guidance or intent

Positive:
First stimulus increases the speed of response to second stimulus

Negative
First stimulus decreases the speed of response to second stimulus. Reaction slower than unprimed.

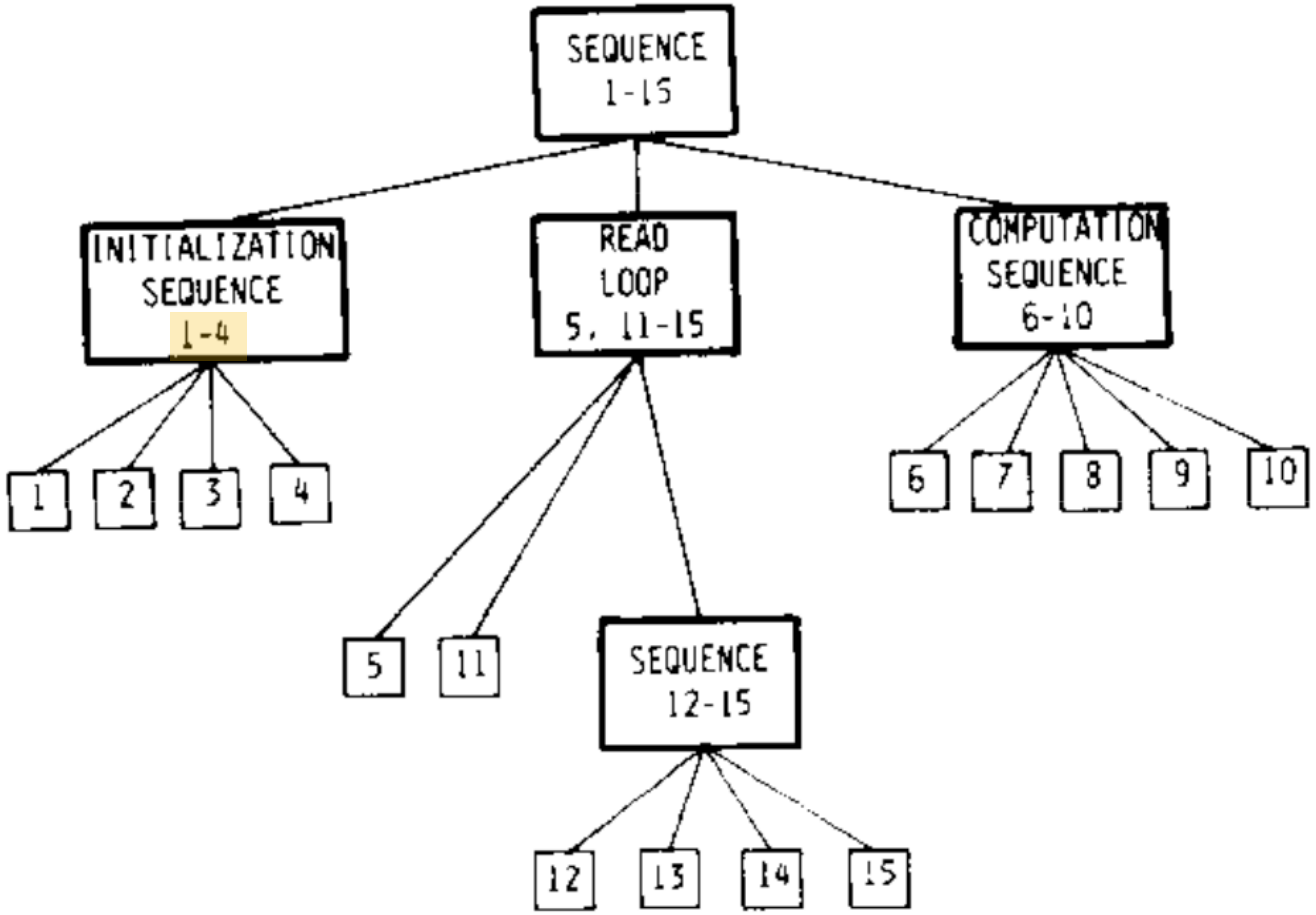
Stimuli that are closely related in an individual’s own mind typically produce positive priming. For more info, take a look at the “spreading activation” literature.

Current theory is that the brain says ‘hey, ignore this category of thing,’ and overriding this earlier instruction makes it take longer.

(McKoon & Ratcliff, 1980, 1984; Ratcliff & McKoon, 1978). In this method, subjects study one or more texts and are subsequently presented with a recognition test in which they must decide whether or not each item in the list was in the text they had just studied. A target item in the test list is preceded in one condition by another item hypothesized to be in the same cognitive unit as the target and thus close in the memory structure. In a second condition the target item is preceded by an item hypothesized to be in a different cognitive unit and thus further away in the memory structure. Under the assumption that activation of an item in the memory structure activates items close to it, especially those in the same cognitive unit, response time to the target preceded by an item in the same cognitive unit should be faster than response time to the same target preceded by an item not in the same cognitive unit (Anderson, 1983; McKoon & Ratcliff, 1980); that is, a priming effect should occur.

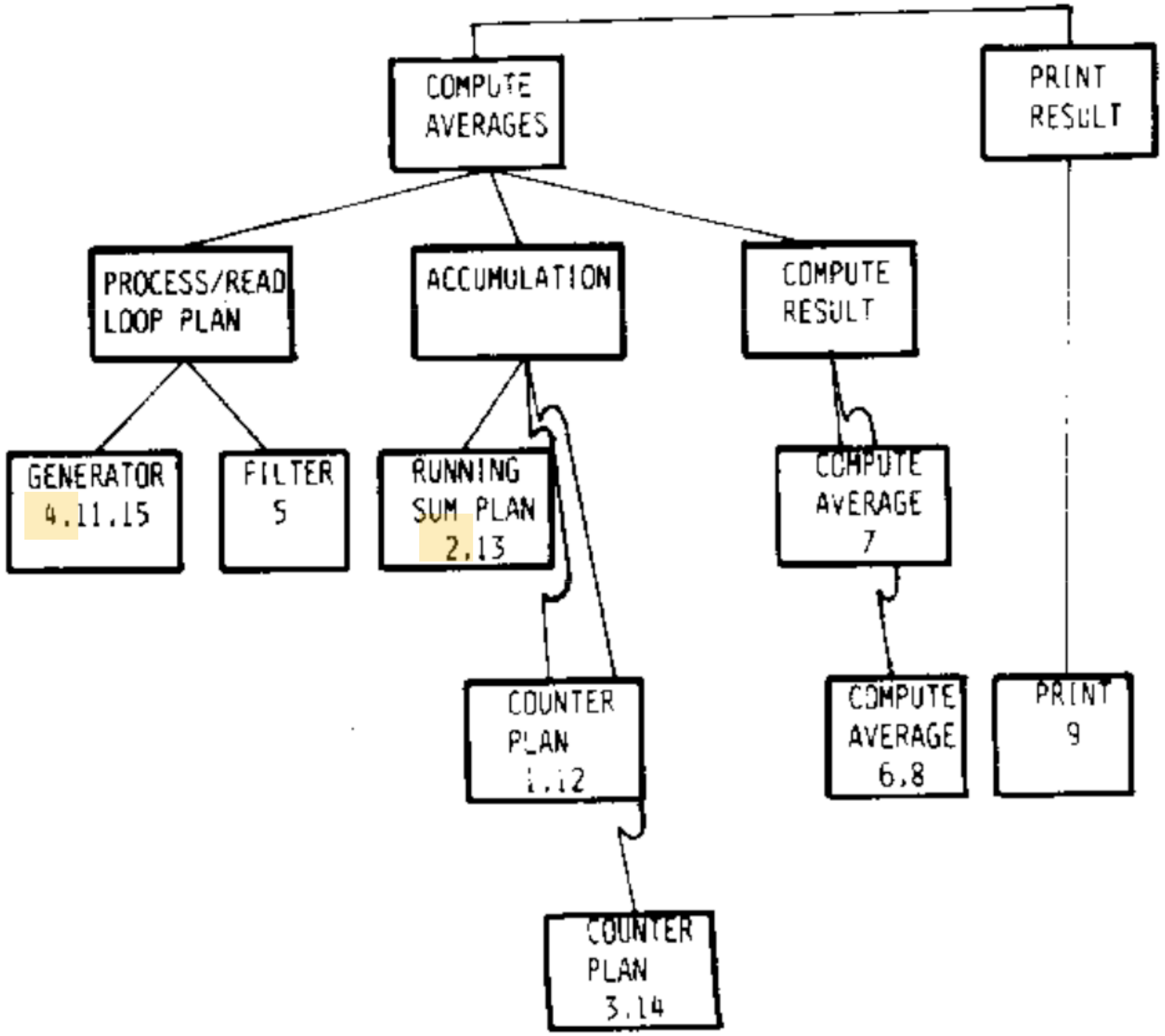
Text Structure (TS)

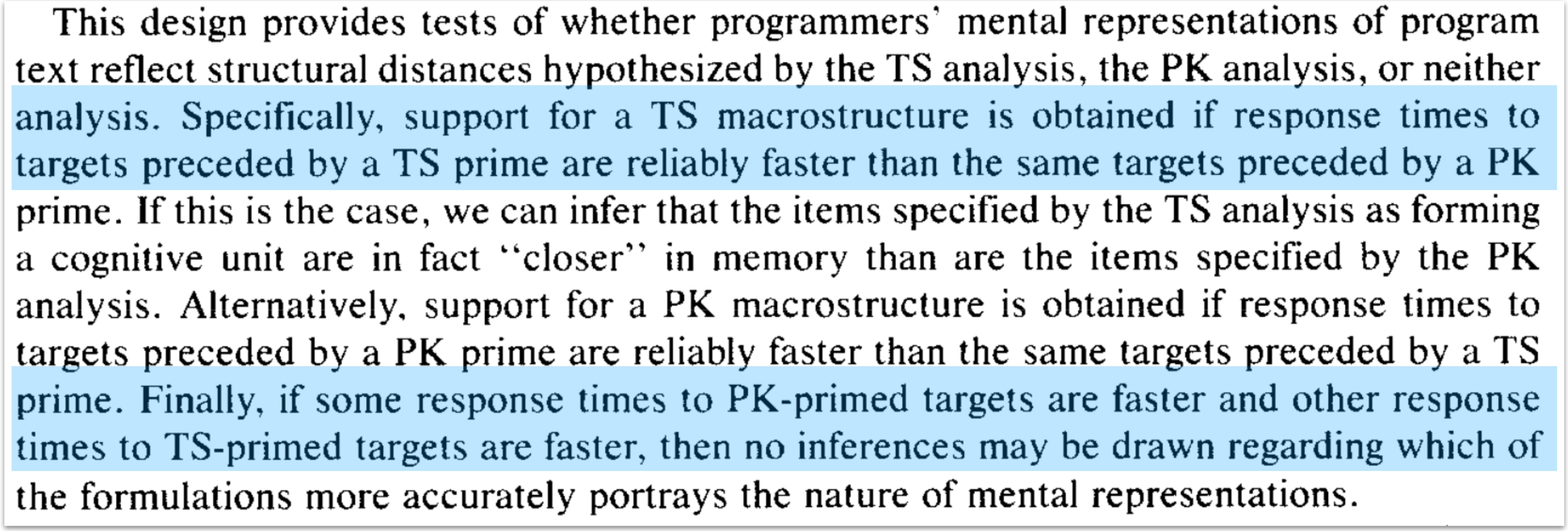
- Basically, control flow



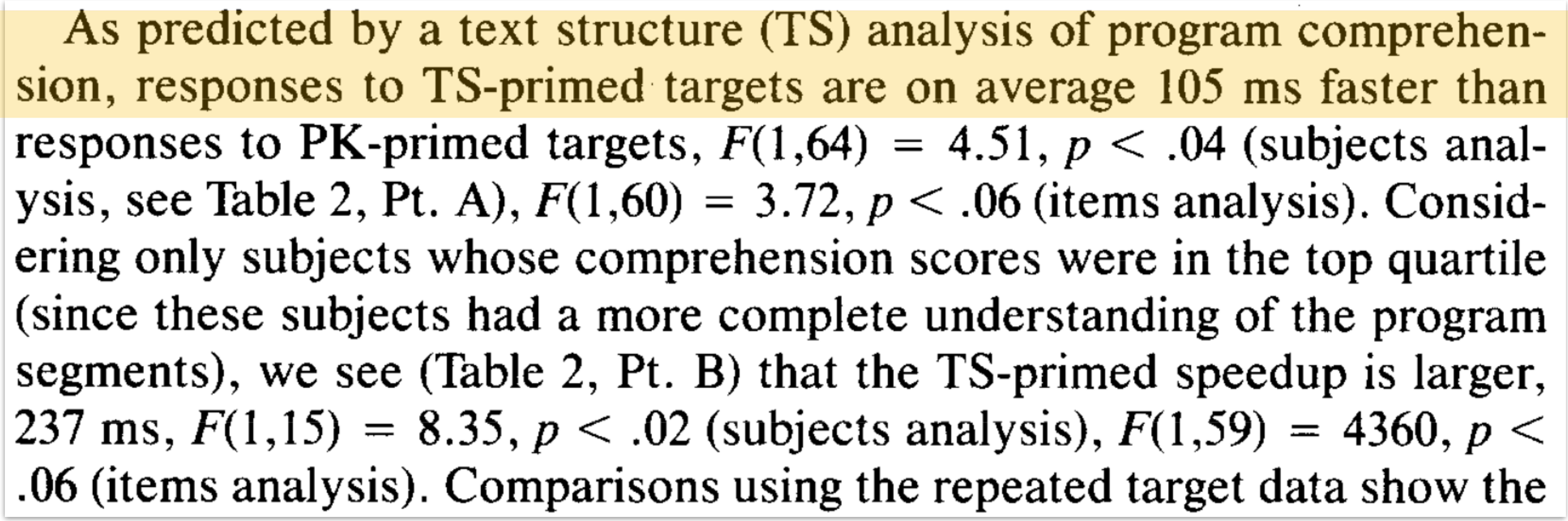
Plan Knowledge (PK)

- Basically, the hierarchical plans we discussed last class, with schemas as nodes





This design provides tests of whether programmers' mental representations of program text reflect structural distances hypothesized by the TS analysis, the PK analysis, or neither analysis. Specifically, support for a TS macrostructure is obtained if response times to targets preceded by a TS prime are reliably faster than the same targets preceded by a PK prime. If this is the case, we can infer that the items specified by the TS analysis as forming a cognitive unit are in fact ‘‘closer’’ in memory than are the items specified by the PK analysis. Alternatively, support for a PK macrostructure is obtained if response times to targets preceded by a PK prime are reliably faster than the same targets preceded by a TS prime. Finally, if some response times to PK-primed targets are faster and other response times to TS-primed targets are faster, then no inferences may be drawn regarding which of the formulations more accurately portrays the nature of mental representations.



As predicted by a text structure (TS) analysis of program comprehension, responses to TS-primed targets are on average 105 ms faster than responses to PK-primed targets, $F(1,64) = 4.51$, $p < .04$ (subjects analysis, see Table 2, Pt. A), $F(1,60) = 3.72$, $p < .06$ (items analysis). Considering only subjects whose comprehension scores were in the top quartile (since these subjects had a more complete understanding of the program segments), we see (Table 2, Pt. B) that the TS-primed speedup is larger, 237 ms, $F(1,15) = 8.35$, $p < .02$ (subjects analysis), $F(1,59) = 4360$, $p < .06$ (items analysis). Comparisons using the repeated target data show the

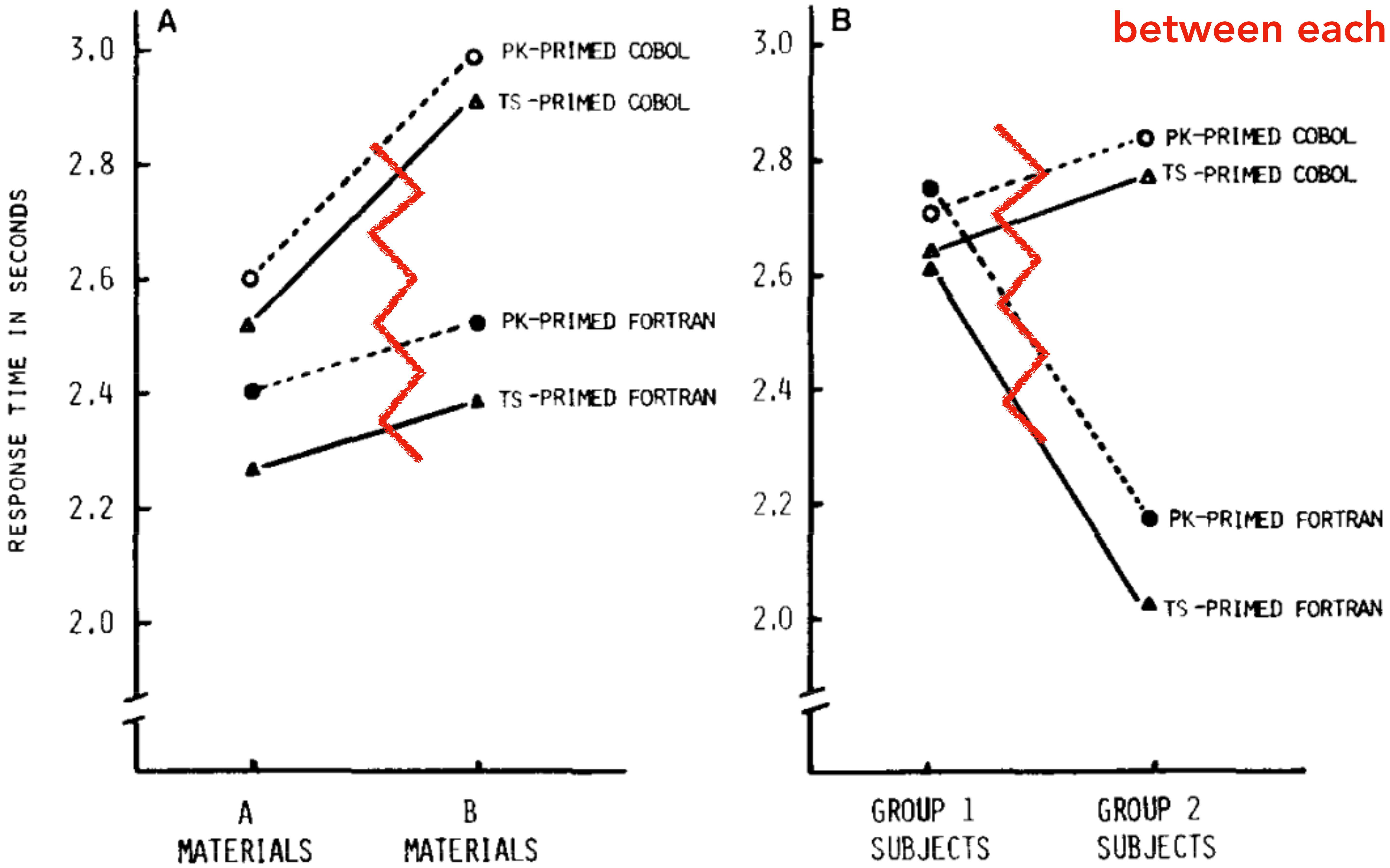


FIG. 7. Study 1 response times for recognition memory items comparing PK-primed item times to TS-primed item times for each set of materials within language adjusted for the effects of subject group (A) and for each subject group within language adjusted for the effects of materials set (B).

Ignore the lines drawn between these groups. I have no idea why they presented the data this way. We just want to look at the pairs of points and the relationship between each pair.

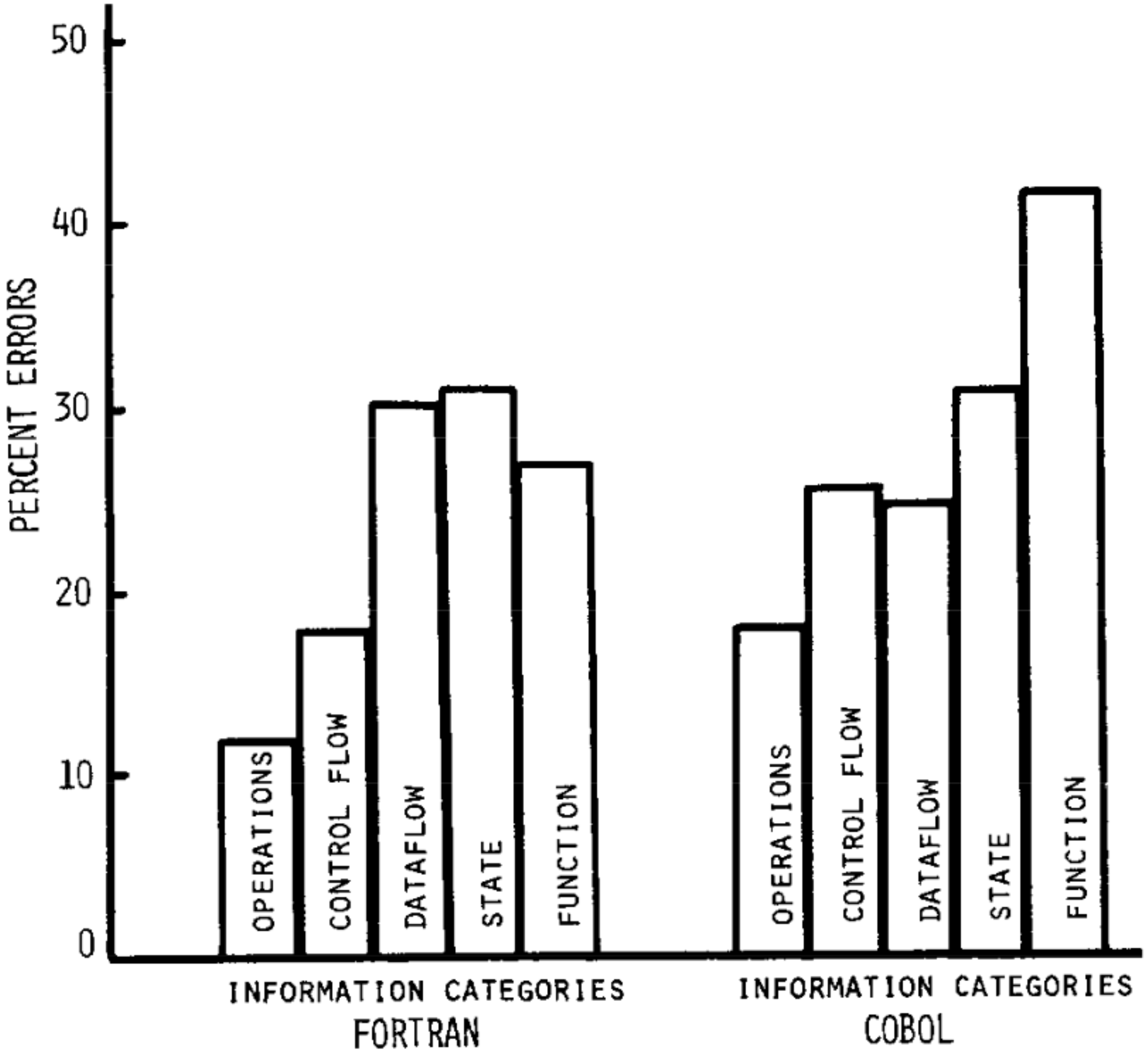


FIG. 8. Study 1 comprehension question error rates by information category for each language.

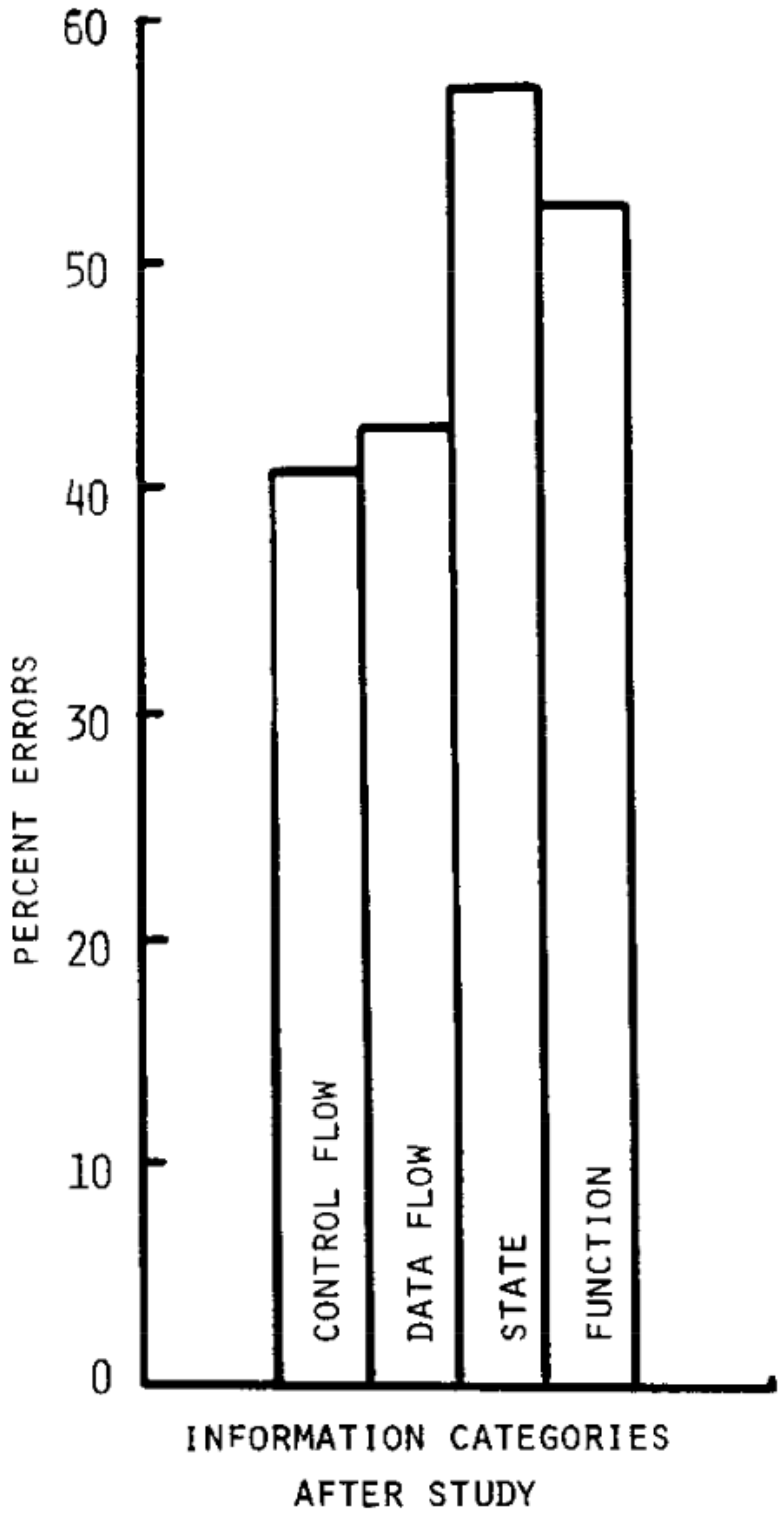
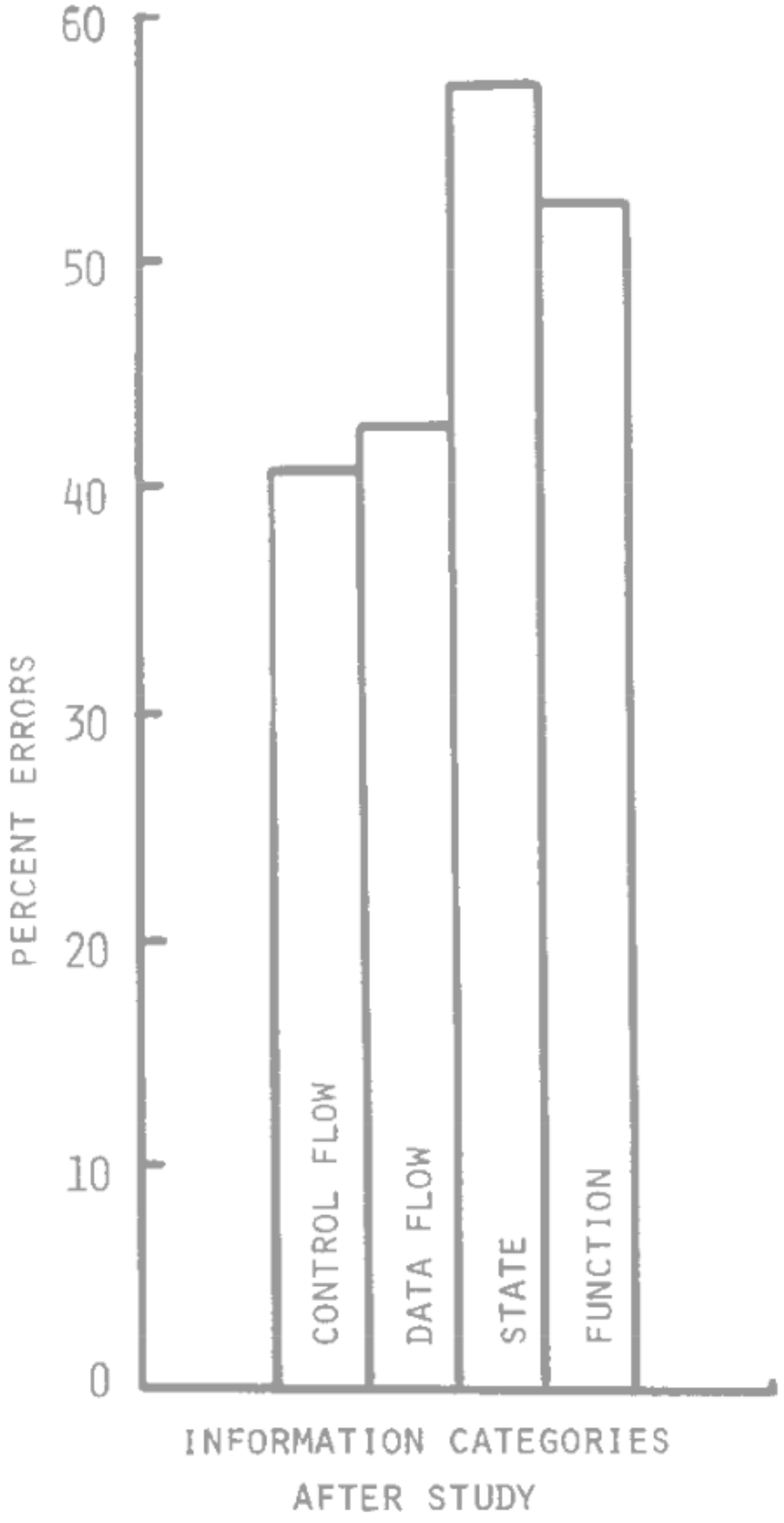
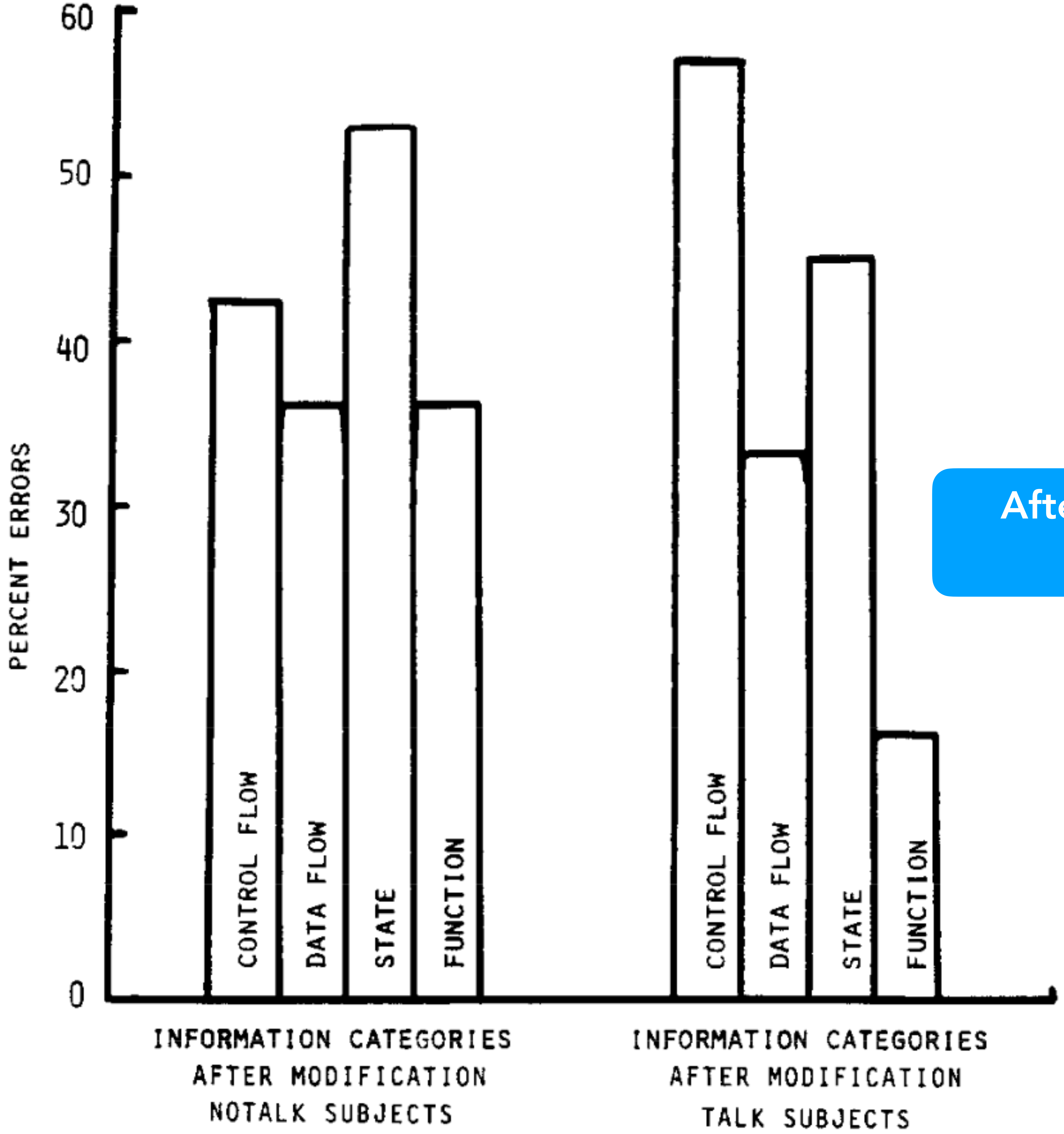


FIG. 10. Study 2 comprehension question error rates by information category, after study task.

After being asked to understand realistic program



hension question error rates by inform



After being asked to *modify* realistic program

FIG. 11. Study 2 comprehension question error rates by information category, after modification task, for talk and notalk subjects.

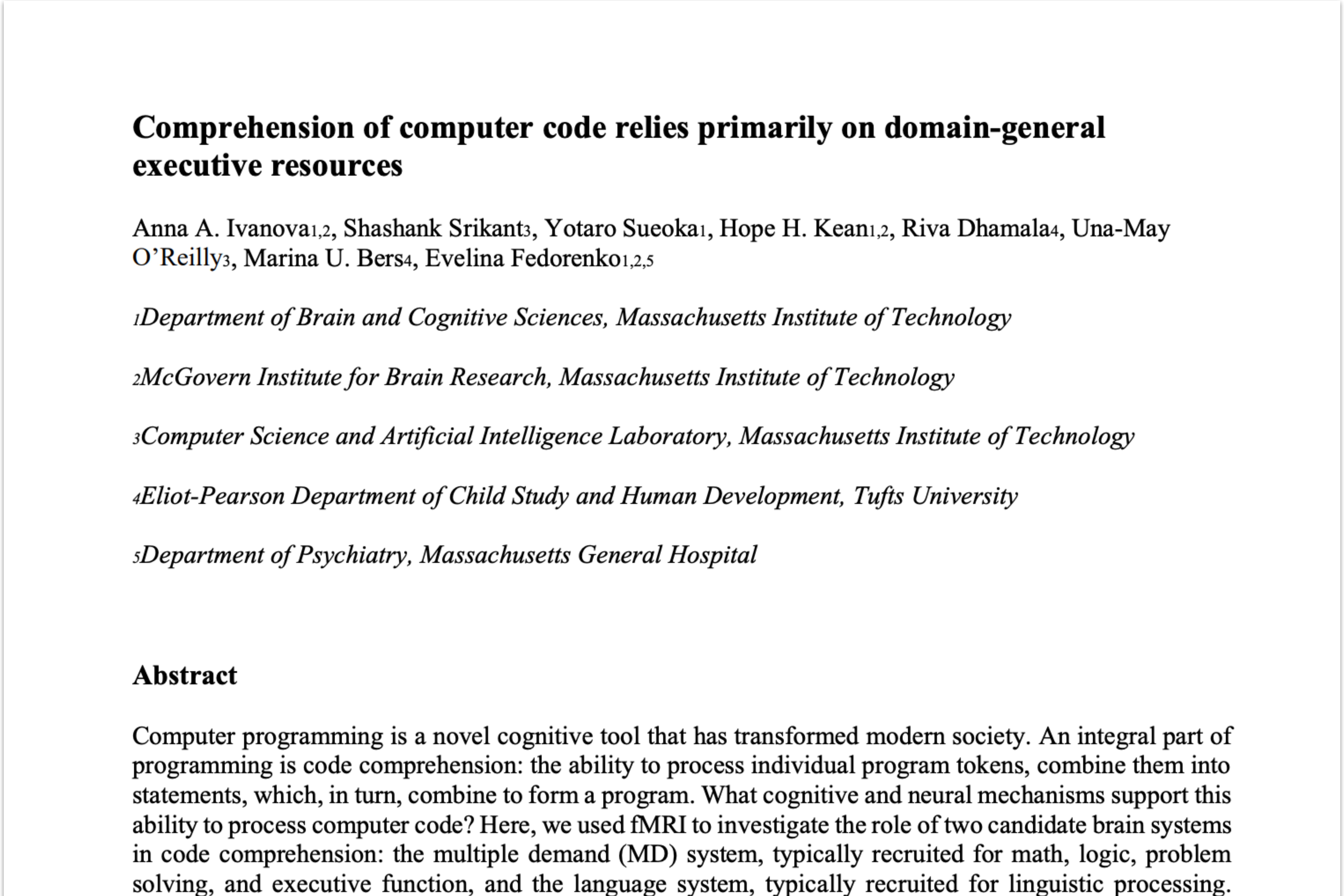
We also found evidence that in *later* stages of program comprehension, under appropriate task conditions, a second representation is available that reflects the functional structure of the program and is expressed in the language of the real world domain to which the program is applied. Our explanations for this later, task-related shift in comprehension are speculative and draw on the concept of a *situation model* representation of the program that is distinct from the macrostructure organization of the textbase (van Dijk & Kintsch, 1983). What is clear from our research is that this second, functional representation is not constructed quickly or automatically. Programmers required extensive involvement with the program before being able to use this structure to respond to questions about the program. Further research is needed to explore the viability of

Reflection

- Did this paper influence your idea of what kinds of information you can get from studies of programmers?

Reflection

- Q: Do you believe a study like this actually tells us anything about the brain internals?



Comprehension of computer code relies primarily on domain-general executive resources

Anna A. Ivanova[1,2], Shashank Srikant[3], Yotaro Sueoka[1], Hope H. Kean[1,2], Riva Dhamala[4], Una-May O'Reilly[3], Marina U. Bers[4], Evelina Fedorenko[1,2,5]

[1]Department of Brain and Cognitive Sciences, Massachusetts Institute of Technology

[2]McGovern Institute for Brain Research, Massachusetts Institute of Technology

[3]Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology

[4]Eliot-Pearson Department of Child Study and Human Development, Tufts University

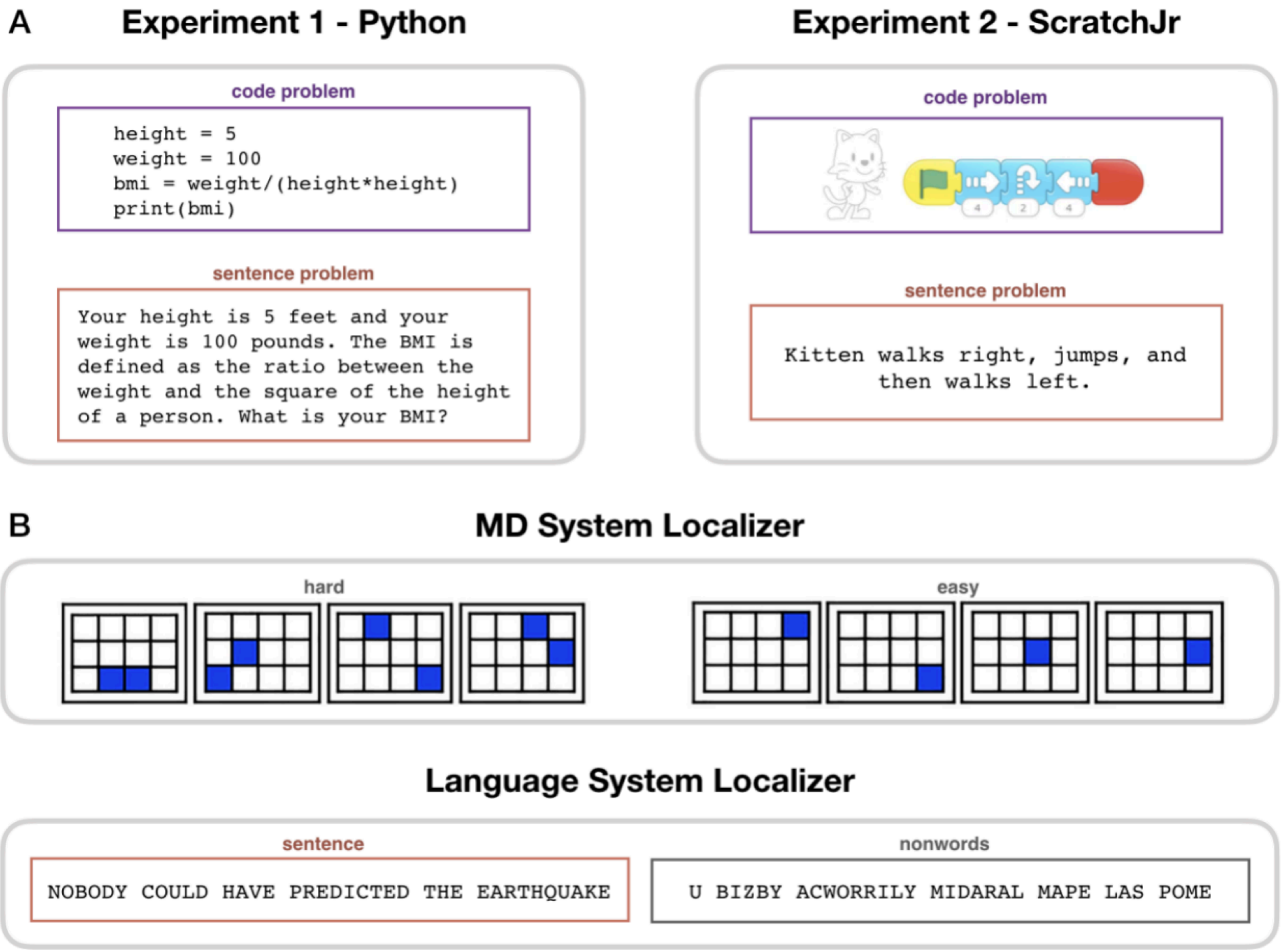
[5]Department of Psychiatry, Massachusetts General Hospital

Abstract

Computer programming is a novel cognitive tool that has transformed modern society. An integral part of programming is code comprehension: the ability to process individual program tokens, combine them into statements, which, in turn, combine to form a program. What cognitive and neural mechanisms support this ability to process computer code? Here, we used fMRI to investigate the role of two candidate brain systems in code comprehension: the multiple demand (MD) system, typically recruited for math, logic, problem solving, and executive function, and the language system, typically recruited for linguistic processing.

Systems
(for our purposes, parts of the brain)

- **Multiple Demand (MD) system**: typically recruited for math, logic, problem solving, executive function
- **Language system**: typically recruited for linguistic processing



```
height = 5
weight = 100
bmi = weight/(height*height)
print(bmi)
```

Your height is 5 feet and your weight is 100 pounds. The BMI is defined as the ratio between the weight and the square of the height of a person. What is your BMI?

Kitten walks right, jumps, and then walks left.

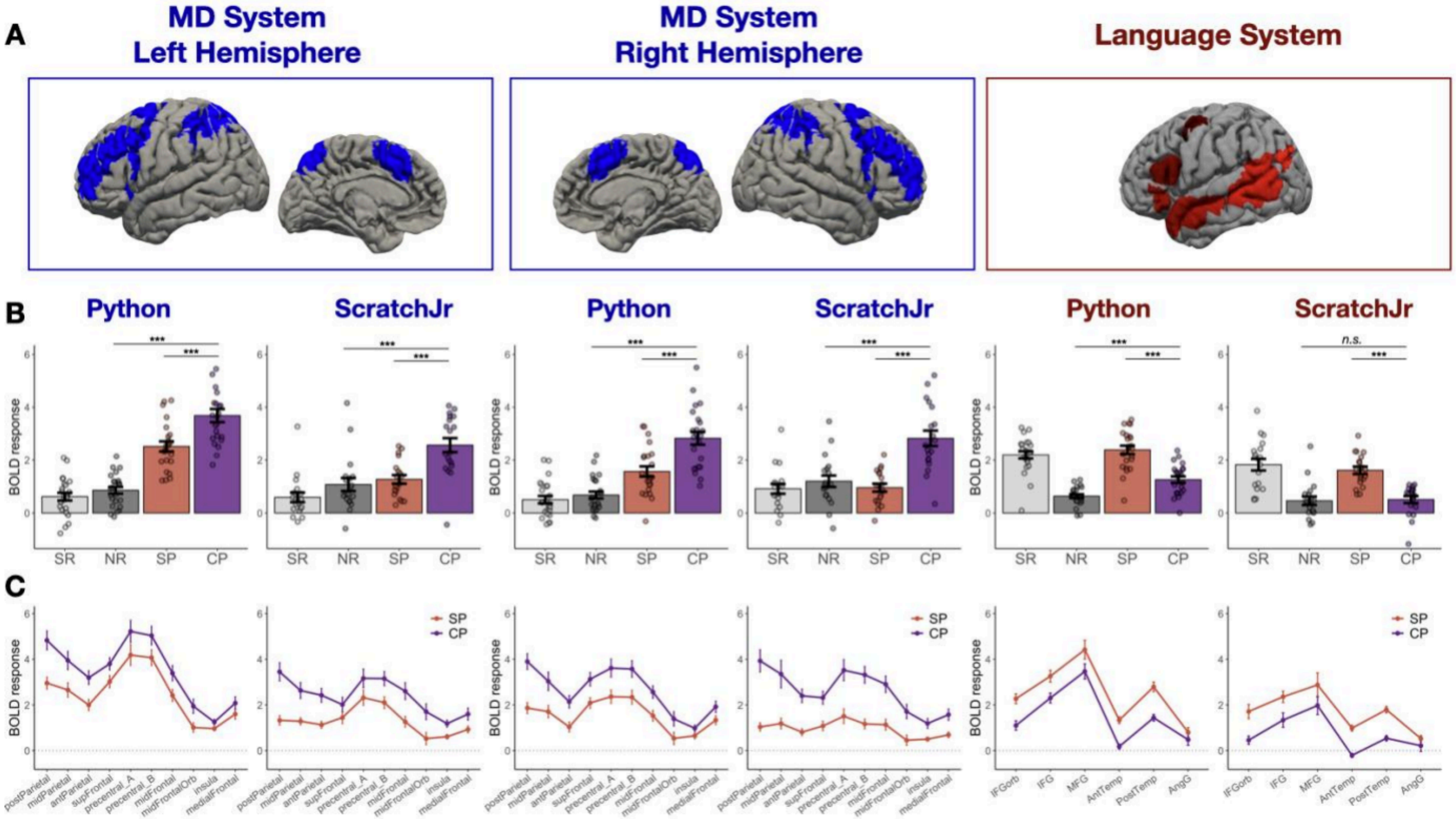
NOBODY COULD HAVE PREDICTED THE EARTHQUAKE

U BIZBY ACWORRILY MIDARAL MAPE LAS POME

Figure 1. *Experimental paradigms. (**A**) Main task. During code problem trials, participants were presented with snippets of code in Python (Experiment 1) or ScratchJr (Experiment 2); during sentence problem trials, they were presented with text problems that were matched in content with the code stimuli. No participants saw both the code and the sentence versions of each problem. (**B**) Localizer tasks. The MD localizer (top) included a hard condition (memorizing positions of 8 blocks appearing two at a time) and an easy condition (memorizing positions of 4 blocks appearing one at a time). The language localizer (bottom) included a sentence reading and a nonword reading condition, with the words/nonwords appearing one at a time.*

Definitely getting recruited for CP, the Code Problem! (purple)

Hm, not getting especially recruited for CP (even with text-based Python)



Broken down by particular brain regions of interest

SR: sentence reading. NR: non-words reading. SP: sentence problem. CP: code problem.

Sometimes you don’t have to invent something fancy—because others have done the hard work of developing measures!

Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research

Sandra G. Hart
Aerospace Human Factors Research Division
NASA-Ames Research Center
Moffett Field. California

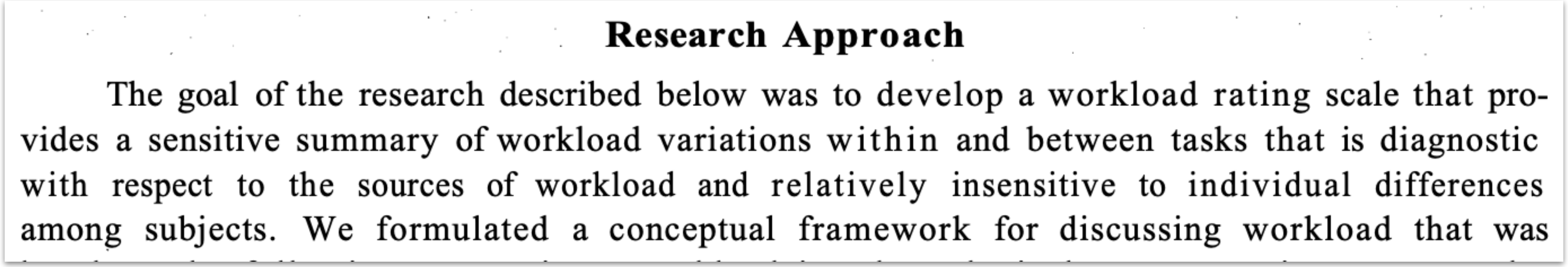
Lowell E. Staveland
San Jose State University
San Jose. California

ABSTRACT

The results of a multi-year research program to identify the factors associated with variations in subjective workload within and between different types of tasks are reviewed. Subjective evaluations of 10 workload-related factors were obtained from 16 different experiments. The experimental tasks included simple cognitive and manual control tasks, complex laboratory and supervisory control tasks, and aircraft simulation. Task-, behavior-, and subject-related correlates of subjective workload experiences varied as a function of difficulty manipulations within experiments, different sources of workload between experiments, and individual differences in workload definition. A multi-dimensional rating scale is proposed in which information about the magnitude and sources of six workload-related factors are combined to derive a sensitive and reliable estimate of workload.

INTRODUCTION

This chapter describes the results of a multi-year research effort aimed at empirically isolating and defining factors that are relevant to subjective experiences of workload and to for-



Research Approach

The goal of the research described below was to develop a workload rating scale that provides a sensitive summary of workload variations within and between tasks that is diagnostic with respect to the sources of workload and relatively insensitive to individual differences among subjects. We formulated a conceptual framework for discussing workload that was

FIGURE 3: RATING SCALE DESCRIPTIONS

Title	Endpoints	Descriptions
OVERALL WORKLOAD	*Low, High*	The total workload associated with the task, considering all sources and components.
TASK DIFFICULTY	*Low, High*	Whether the task was easy or demanding, simple or complex, exacting or forgiving.
TIME PRESSURE	*None, Rushed*	The amount of pressure you felt due to the rate at which the task elements occured. Was the task slow and leisurely or rapid and frantic?
PERFORMANCE	*Failure, Perfect*	How successful you think you were in doing what we asked you to do and how satisfied you were with what you accomplished.
MENTAL/SENSORY EFFORT	*None, Impossible*	The amount of mental and/or perceptual activity that was required (e.g., thinking, deciding, calculating, remembering, looking, searching, etc.).
PHYSICAL EFFORT	*None, Impossible*	The amount of physical activity that was required (e.g., pushing, pulling, turning controlling, activating, etc.).
FRUSTRATION LEVEL	*Fulfilled, Exasperated*	How insecure, discouraged, irritated, and annoyed versus secure, gratified, content, and complacent you felt.
STRESS LEVEL	*Relaxed, Tense*	How anxious, worried, uptight, and harresed or calm, tranquil, placid, and relaxed you felt.
FATIGUE	*Exhausted, Alert*	How tired, weary, worn out, and exhausted or fresh, vigorous, and energetic you felt.
ACTIVITY TYPE	*Skill Based,* *Rule Based,* *Knowledge Based*	The degree to which the task required mindless reaction to well-learned routines or required the application of known rules or required problem solving and decision making.

Discussion: Programming, Programmers, and Experiments

- Humans and human-involved activities are messy!
    - Does every project have to involve experiments, be reduced to something small that can be randomized or something easy to measure?
        - No!
- So what’s the big deal about all these ways of studying programmers, programming?
    - One more tool in our toolbox
    - Gives us a way to make claims about:
        - Usability
        - Programmer behavior
        - Programmer needs
        - Programmer workload
        - Programmer mental models
    - …that doesn’t rely on reviewers thinking the same way we do
    - A way to avoid some big mistakes, by doing a bit more work upfront
        - Take a look back at PL history

The plan for next class week!

- Next class week, we're taking a tour of PL + HCI!
- Instead of a standard lectures, we'll have mini-presentations on research works
- …which means instead of a normal HW assignment, you'll put together a 4-minute mini-presentation on a research paper or project
    - These will be very casual! Low-stakes presentations. Just give us a sense of what the work is doing and how. :)
    - You can pick any work that combines PL and HCI, but there's a list of exciting papers available if you want some hints
- More details are in next week's Assignment pdf
- Once you've picked a paper, please *scan the schedule to make sure no one else has already claimed the paper*! Then sign yourself up for a slot so no else claims your paper first. :)

Activity

- What are 3 things you wish you knew about programmers' internal state?

Activity

- Pick one of your questions and brainstorm a study design that could help you answer it.