CS 338: Graphical User Interfaces

Lecture 7-2: Intelligent Interfaces

Intelligent interfaces

- What is not an intelligent interface?
  - any "intelligent system"
    - something that strives to pass the Turing Test
    - e.g., expert system
    - e.g., neural network

    → doesn't necessarily include an interface
  - any "good interface"
    - that is, a well-designed interface

    → doesn't necessarily include intelligence

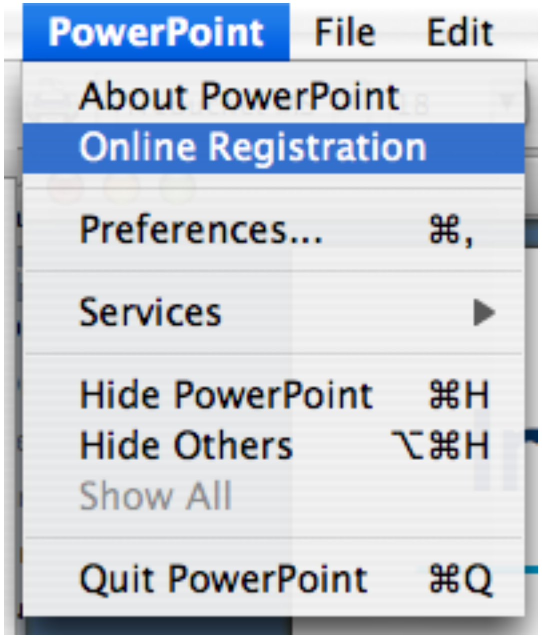
Thanks to Annika Wærn for material from her document "What is an intelligent interface?"

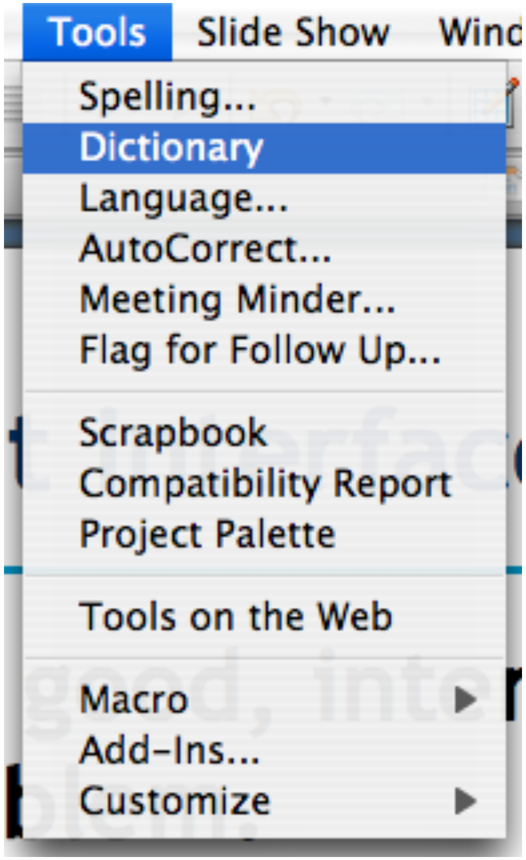
Intelligent interfaces

- Today's intelligent interfaces utilize...
  - user adaptivity to change behavior for different users and situations
  - user modeling to maintain dynamic knowledge about user knowledge, preferences, etc.
  - natural language technology to interpret and/or generate text and/or speech
  - dialog modeling to maintain a dynamic multimodal interaction with the user
  - explanation generation to ensure that the user understands what is happening

Intelligent interfaces

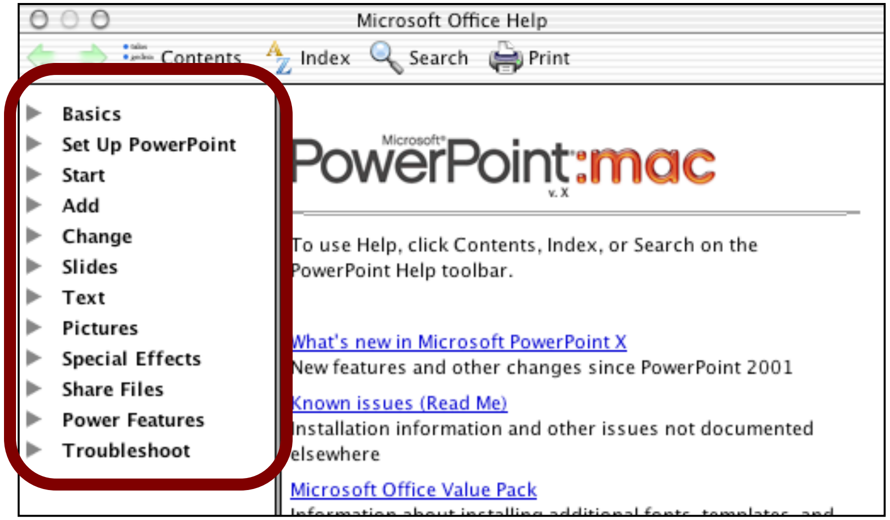
- Creating good, interactive intelligence is challenging.
- Let's say we have a menu of options…
- How might this adapt in an intelligent way?



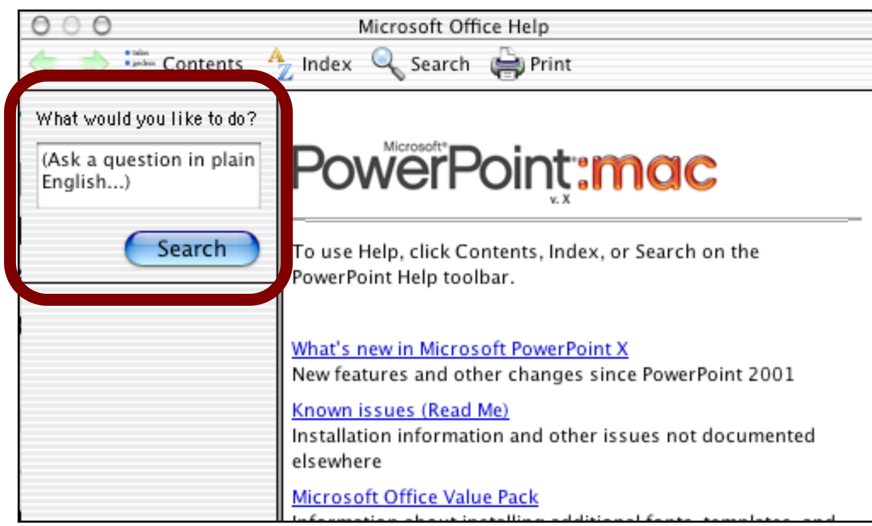


Example: Intelligent help

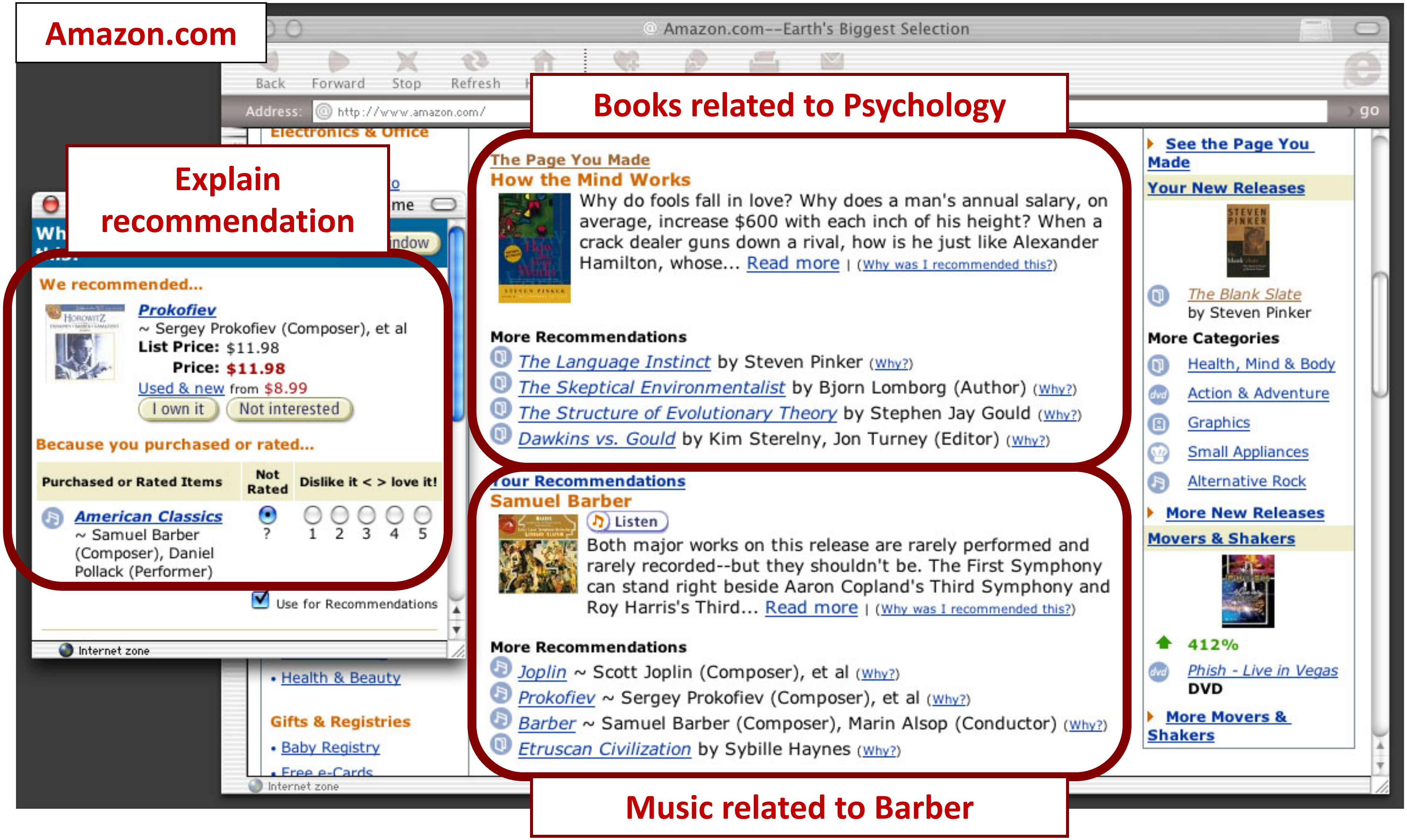
- Help doesn't have to be intelligent — e.g., searchable index / table of contents



- Intelligent help can consider context or provide human-like dialog for searching

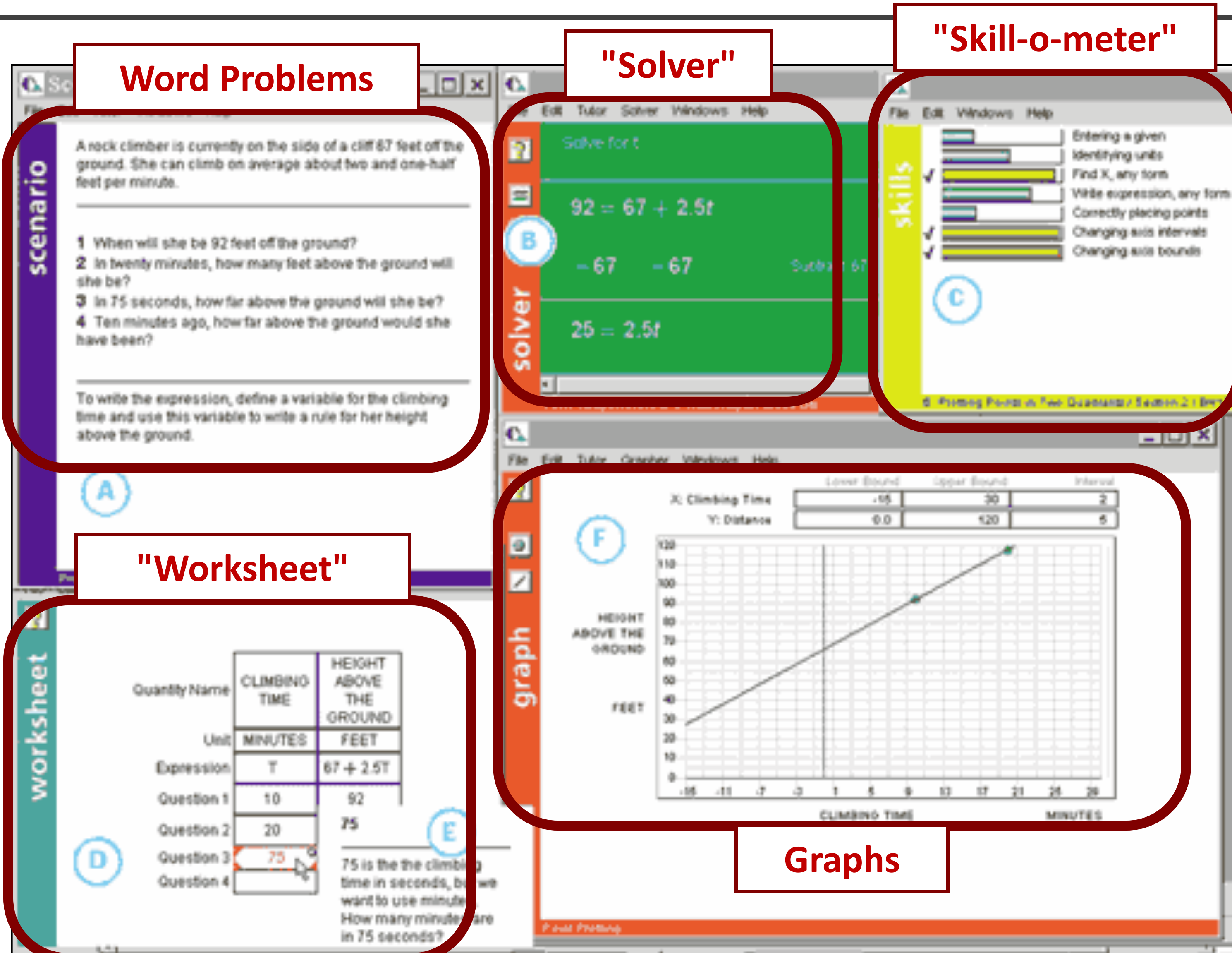


Example: Information filtering



Example: Intelligent tutoring

Carnegie Learning: Cognitive Tutor Algebra I



Intelligent tutoring

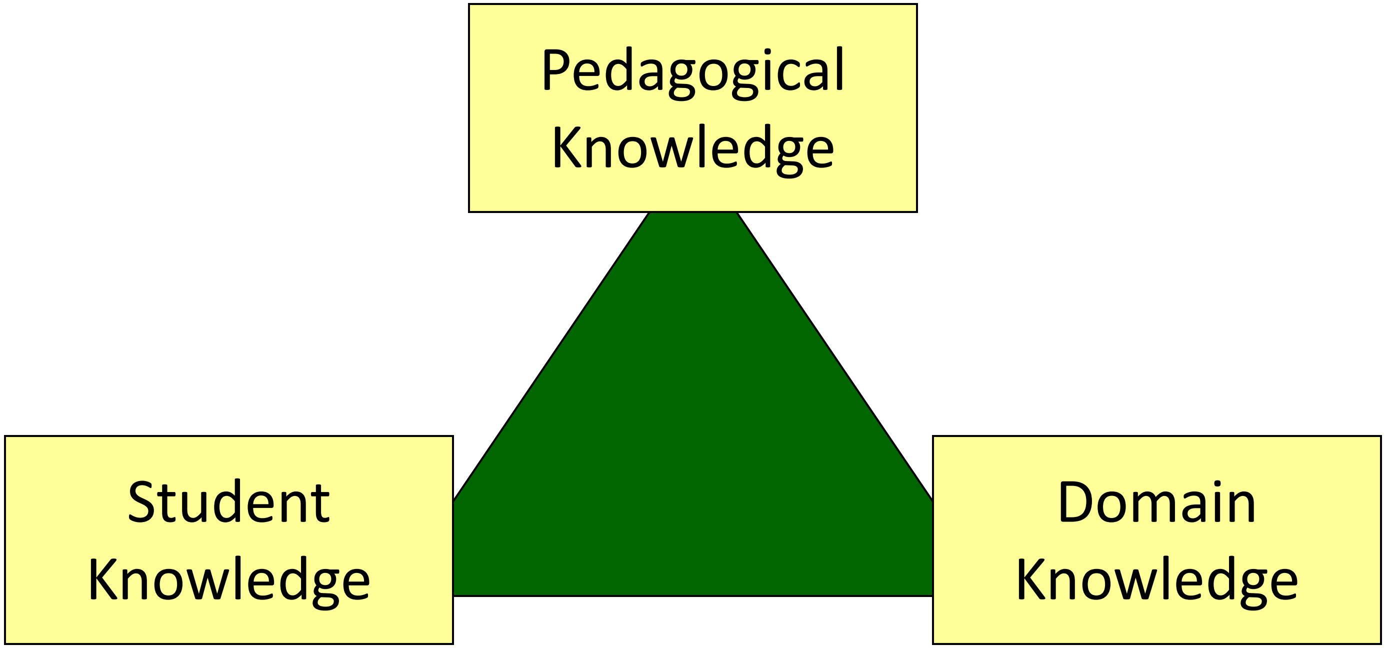
- The "two-sigma effect" of human tutoring

> Students who receive one-on-one instruction perform *two standard deviations better* than students who receive traditional classroom instruction.

- Advantages of a computer tutor
  - a teacher for every student? not feasible.
    a computer tutor for every student? sure!
  - enables practice, practice, practice on possibly large (even infinite) database of problems
  - but can also answer questions, provide guidance
  - more "human-like," personalized instruction than a simple computer practice application

Computer tutor components

- What makes up a computer tutor?



Computer tutor components

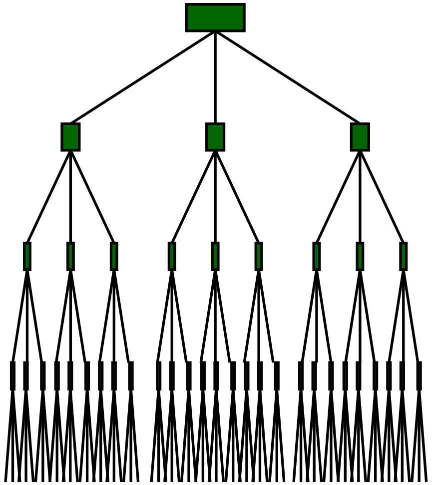
- But specifically, a computer tutor can be boiled down to 2 main components:
  - (1) User model
    - computational representation for what the student knows and doesn't know
  - (2) Model tracing
    - update the student model by associating observed student actions with model predictions

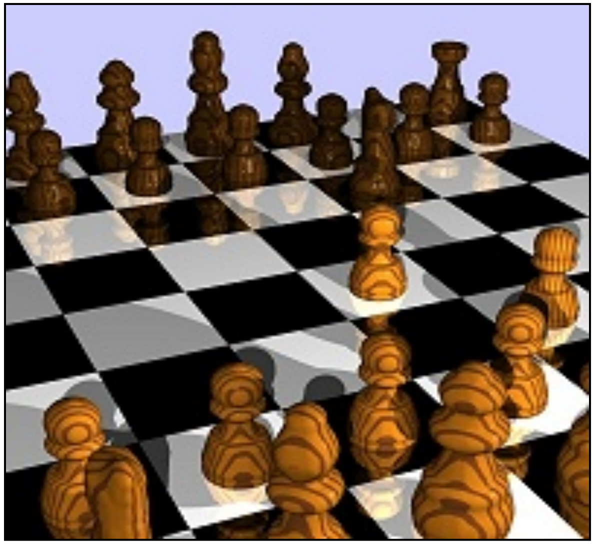
(1) User model

- How can we represent student intelligence?
- Lots of programs do "intelligent" things
  - add really big numbers really quickly
  - diagnose patient symptoms
  - predict earthquakes, volcanic eruptions
  - etc., etc., etc. ...
- Typically, they don't do things like humans
  - sometimes, methods inspired by humans
    - e.g., neural networks
  - but usually, the goal is to solve a problem, not represent human knowledge

Example: Playing chess

- Computers search lots and lots of possible moves and their implications





- This can beat the best human players.
- But how do humans play chess?

Example: Sorting numbers

- Please write down these numbers:

23	**78**	**32**	**44**	**52**	**17**	**89**	**41**

- Now sort them in increasing order…

Example: Sorting numbers

- How did you do it?

17	**23**	**32**	**41**	**44**	**52**	**78**	**89**

- Computers: many efficient algorithms
  - you know the Quicksort algorithm
- Humans (often): find smallest, write it, find 2nd smallest, write it, …
  - not nearly as efficient as Quicksort!
  - so why don't we use Quicksort?
    too many things to track, especially in your head

Creating a human-like user model

- Humans have many limitations
  - cognition (e.g., recall 7±2 digits)
  - perception (e.g., the eye's fovea)
  - attention (e.g., driving & dialing)
  - behavior (e.g., typing speed)
- For machines to think & act like people, they need to incorporate these limitations.
  - are you telling me we should build machines that make mistakes?!?
  - well, they don't need to make mistakes, just explain and predict them

Cognitive architectures

- Frameworks for building computer models of how people (e.g., students) think and act
- Serve two purposes
  - explain cognition with a psychological theory
  - simulate cognition with computer models
- Incorporate human abilities & limitations!
  - abilities: learning & speedup, pattern matching, …
  - limitations: memory decay, physical constraints, …

Categories of cognitive architectures

- Symbolic
  - representation based on symbols & relations
    - e.g., Harrisburg, capital, Pennsylvania
- Connectionist
  - representation based on connected nodes
    - e.g., neural networks
- Hybrid
  - some mix of symbolic and connectionist
  - what most architectures strive for today

Note: These categories are very broad and there's lots of gray area!

Production system architectures

- Key: Represent skills as production rules (or simply "productions")
- Production = conditions + actions
  - when conditions are satisfied, actions "fire"

IF	my goal is to adjust heating & temperature < 70° F
THEN	turn on heating

IF	my goal is to adjust heating & temperature > 70° F
THEN	turn off heating

- Advantages of production systems
  - parallel and serial processing
  - independence of production rules
  - interruptible and flexible control

Components of a production system

- Declarative knowledge
  - consists of "chunks"
  - simple facts: "2+3=5"
  - knowledge of situation: "Mary is in front of me"
  - current and past goals: "add two numbers"
    - goal stack: push & pop
- Procedural knowledge
  - consists of productions
  - production = condition-action rule
  - actions can act upon physical world or change memory contents

Declarative knowledge

- Chunks comprising:
  - chunk type: what kind of chunk is it?
  - slots: what information does the chunk contain?
  - slot values: what are the actual units in the slots?

```
TWO
    isa number

THREE
    isa number

FIVE
    isa number
```

```
TWO+THREE
    isa        plus-fact      ← chunk type
    addend1    TWO
    addend2    THREE
    sum        FIVE
```

slots

slot values

Procedural knowledge

- Productions (rules) comprising:
  - left-hand side = matching: goal + other chunks
  - right-hand side = acting: motor + memory

```
ADD-NUMBERS

IF        current goal is to add n1 and n2
          and we can retrieve a plus-fact for n1 and n2

THEN      get the sum in the plus-fact
          and type the sum
```

Production system example

Declarative Knowledge

GOAL	
isa	add-numbers
TWO+THREE	
isa	plus-fact
addend1	TWO
addend2	THREE
sum	FIVE
FIVE+SIX	
isa	plus-fact
addend1	FIVE
addend2	ONE
sum	ELEVEN

…

Procedural Knowledge

ADD-COLUMN<10

IF	goal is to add numbers
	& right-most column is n1 & n2
	& retrieve sum for n1+n2, & carry
	& sum < 10
THEN	write sum in column
	& set carry to 0

ADD-COLUMN>=10

IF	goal is to add numbers
	& right-most column is n1 & n2
	& retrieve sum for n1+n2, & carry
	& sum >= 10
THEN	write sum-10 in column
	& set carry to 1

DONE-ADDITION-CARRY=0 …

DONE-ADDITION-CARRY=1 …

Production system example

Problem

25
+36

Trace

ADD-COLUMN>=10
--> retrieves 5+6=11
--> writes 1, sets carry to 1

ADD-COLUMN<10
--> retrieves 2+3=5, adds 1
--> writes 6, sets carry to 0

DONE-ADDITION-CARRY=0

--> done problem!

Procedural Knowledge

ADD-COLUMN<10

IF	goal is to add numbers
	& right-most column is n1 & n2
	& retrieve sum for n1, n2, & carry
	& sum < 10
THEN	write sum in column
	& set carry to 0

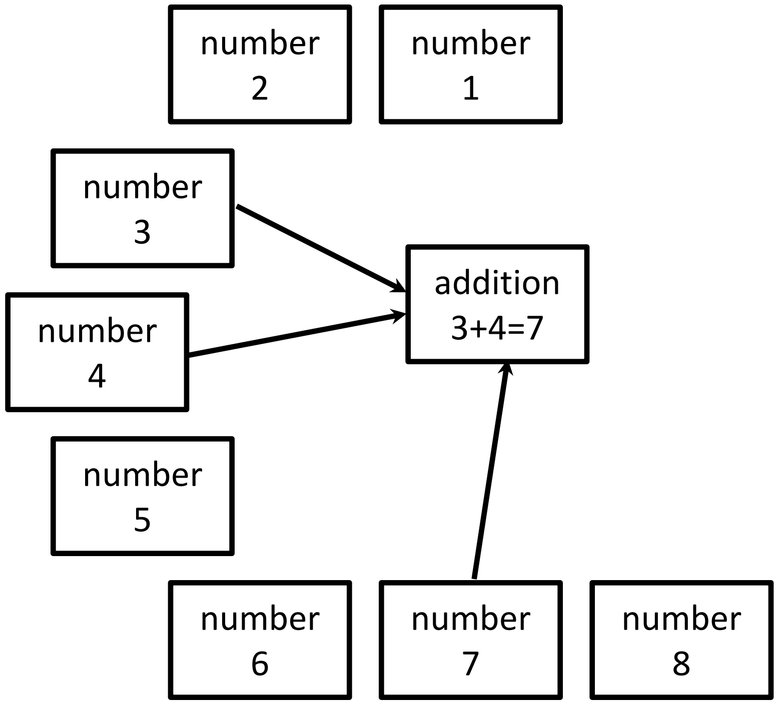
ADD-COLUMN>=10

IF	goal is to add numbers
	& right-most column is n1 & n2
	& retrieve sum for n1, n2, & carry
	& sum >= 10
THEN	write sum-10 in column
	& set carry to 1

DONE-ADDITION-CARRY=0 …
DONE-ADDITION-CARRY=1 …

Theory of Memory

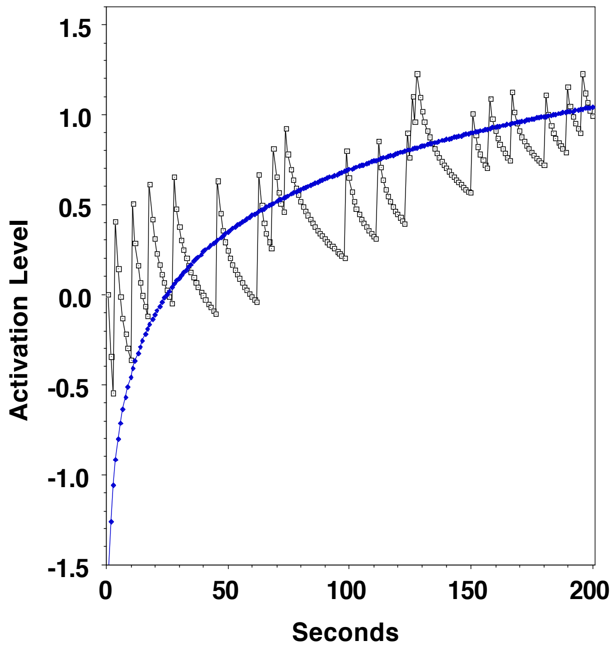
- Memory is a network of information "chunks"
  - chunks are represented computationally as "symbols", as in traditional AI
  - chunks / symbols can be thought of as abstractions of neural patterns in the brain

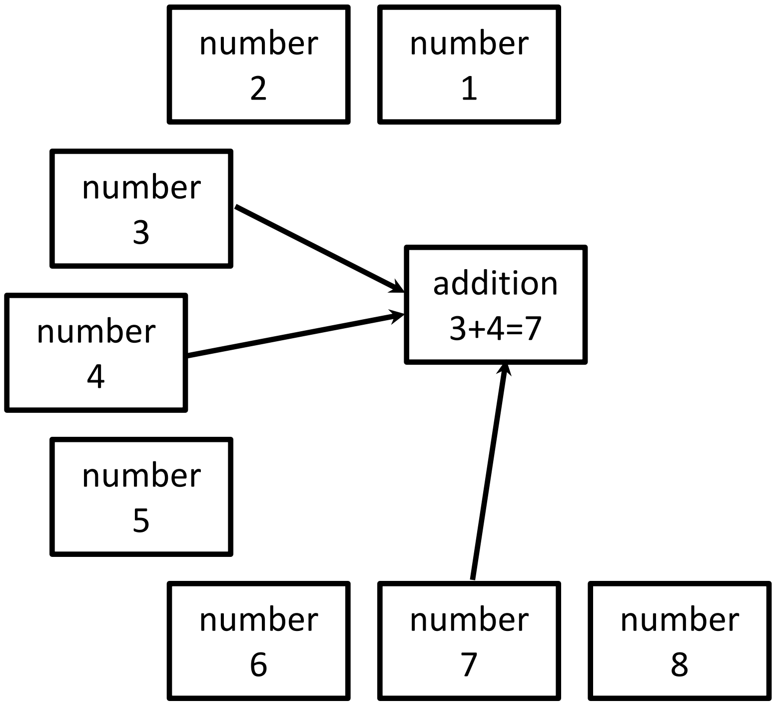


Theory of Memory

- Memory recall is dependent on...
  - prior use/rehearsal of chunk

$$B_i = \ln\left(\sum_{k=1}^{n} t_k^{-d}\right)$$





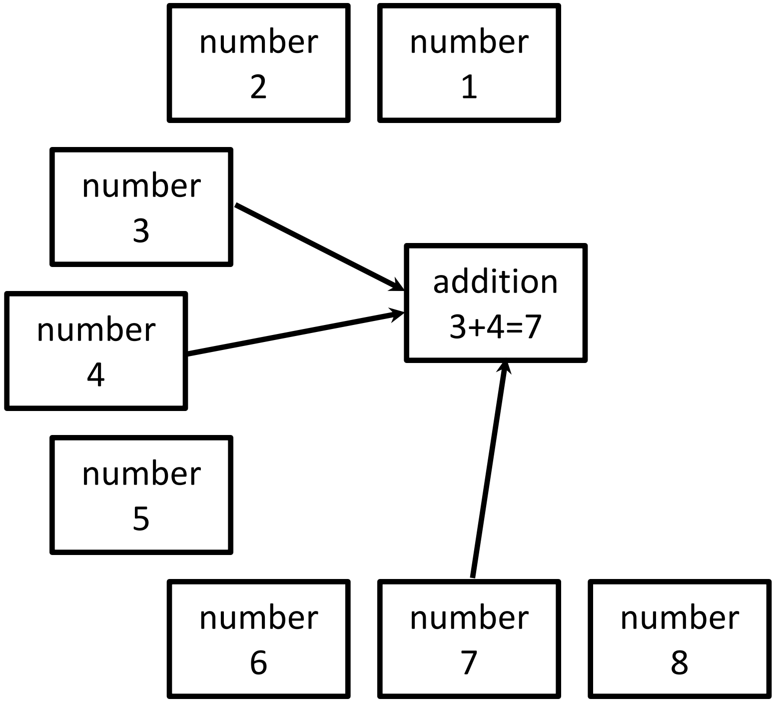
Theory of Memory

- Memory recall is dependent on...
  - prior use/rehearsal of chunk

$$B_i = \ln\left(\sum_{k=1}^{n} t_k^{-d}\right)$$

  - presence of context

$$A_i = B_i + \sum_{j \in C} W_j S_{ji}$$



Theory of Memory

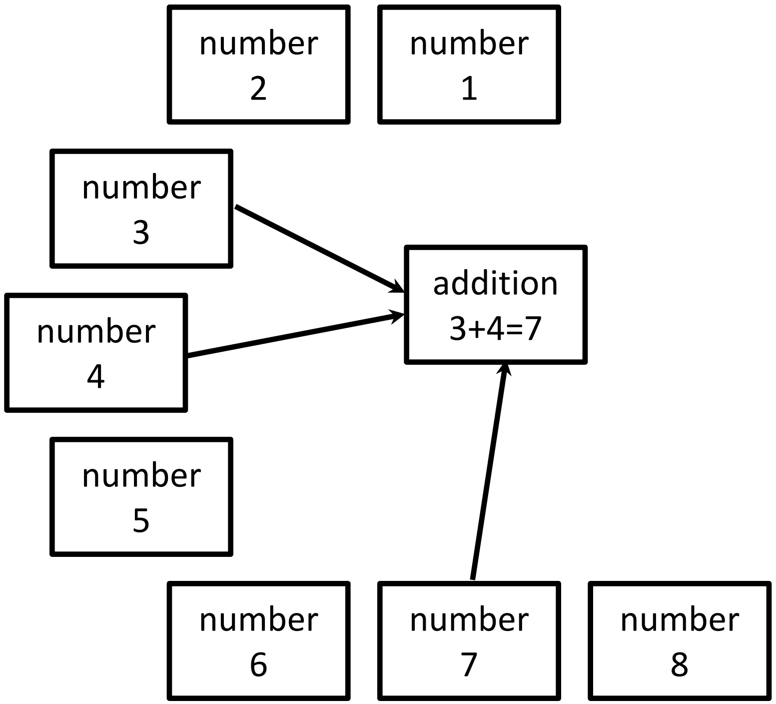
- Memory recall is dependent on...
  - prior use/rehearsal of chunk

$$B_i = \ln\left(\sum_{k=1}^{n} t_k^{-d}\right)$$

  - presence of context

$$A_i = B_i + \sum_{j \in C} W_j S_{ji}$$

  - to compute Prob(recall) and time to recall...



$$Time = Fe^{-A_i} \qquad Prob = 1/\left(1 + e^{-(A_i - \tau)/s}\right)$$

Limitations: Symbolic knowledge

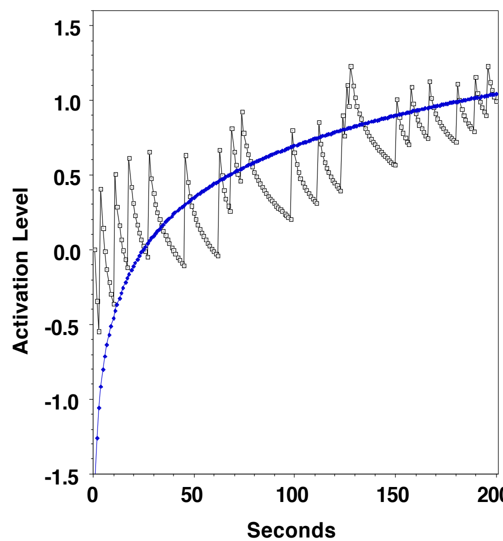
- The most basic limitation is when the model doesn't know something
  - declarative: absence of factual chunks
  - procedural: absence of skills
- Overcoming the limitation
  - acquiring new knowledge through…
    - direct perception
    - indirect inference / reasoning
      (analogy, metaphor, discovery, etc.)
  - compilation of declarative → procedural
    - e.g., driving with a stick-shift

Limitations: Subsymbolic parameters

- Typically, we have "subsymbolic" parameters (i.e., numeric values)
  associated with symbols
- Example: Declarative chunk "activation"
  - represents how easily chunk can be remembered
  - changes over time with learning

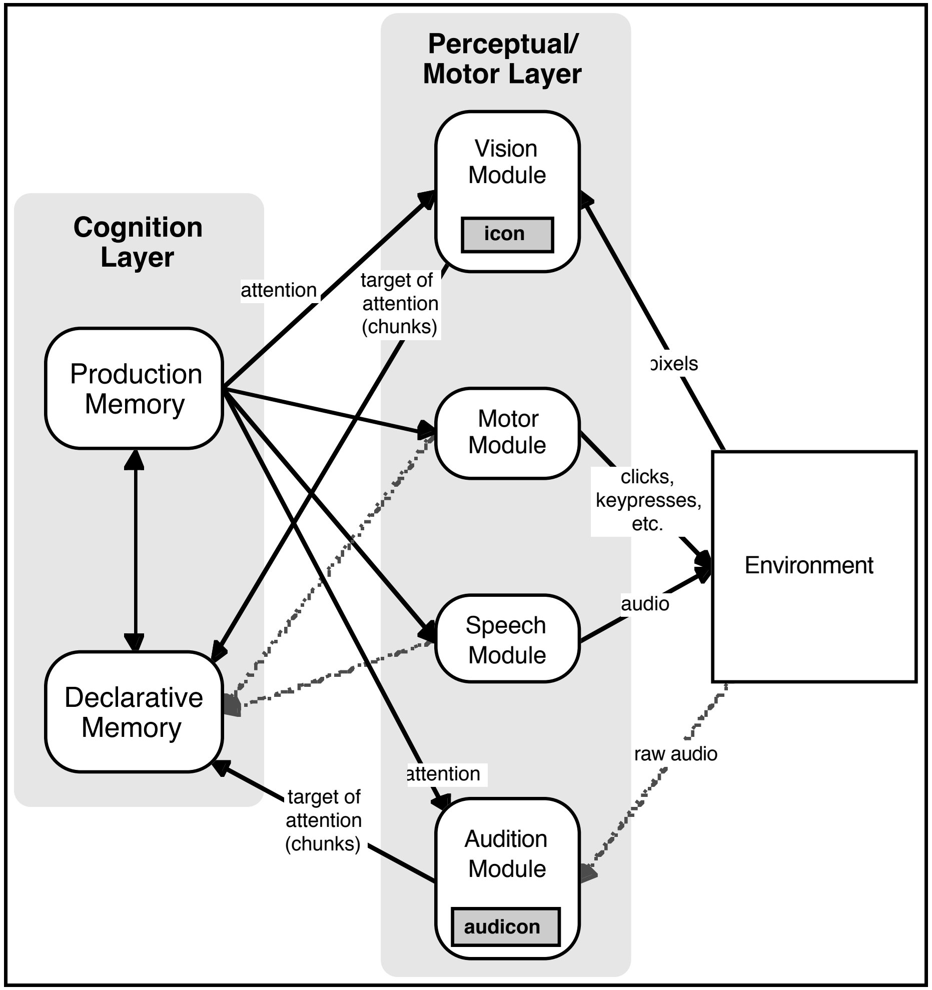
```
TWO+THREE
    isa         plus-fact
    addend1     TWO
    addend2     THREE
    sum         FIVE
```

Activation = 3.24



Limitations: Perceptual-motor

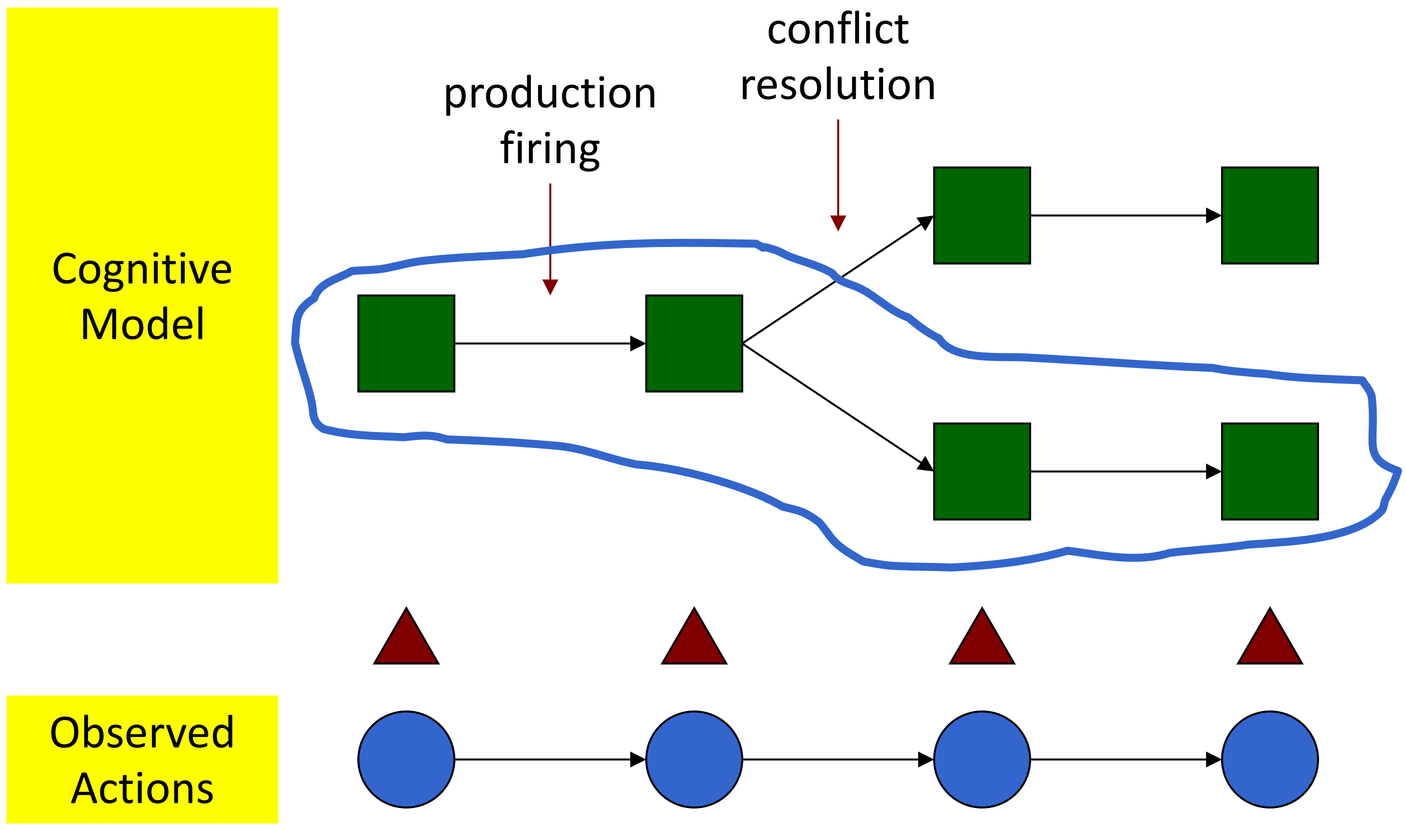
- Modules allow model to interact with a (real or simulated) external environment
- Modules include performance parameters & very limited parallelism



(2) Model tracing

- Model tracing = relating observed actions with hidden cognitive states
- In essence, "think" along with student and keep track of the cognitive state
- From a given point in time...
  - simulate all possible "thought" sequences (i.e., all possible firings for productions)
  - determine which sequence best matches the actions observed from the student
  - make the best matching sequence the current "estimated" cognitive state
- What is "cognitive state" in this case?

Model tracing

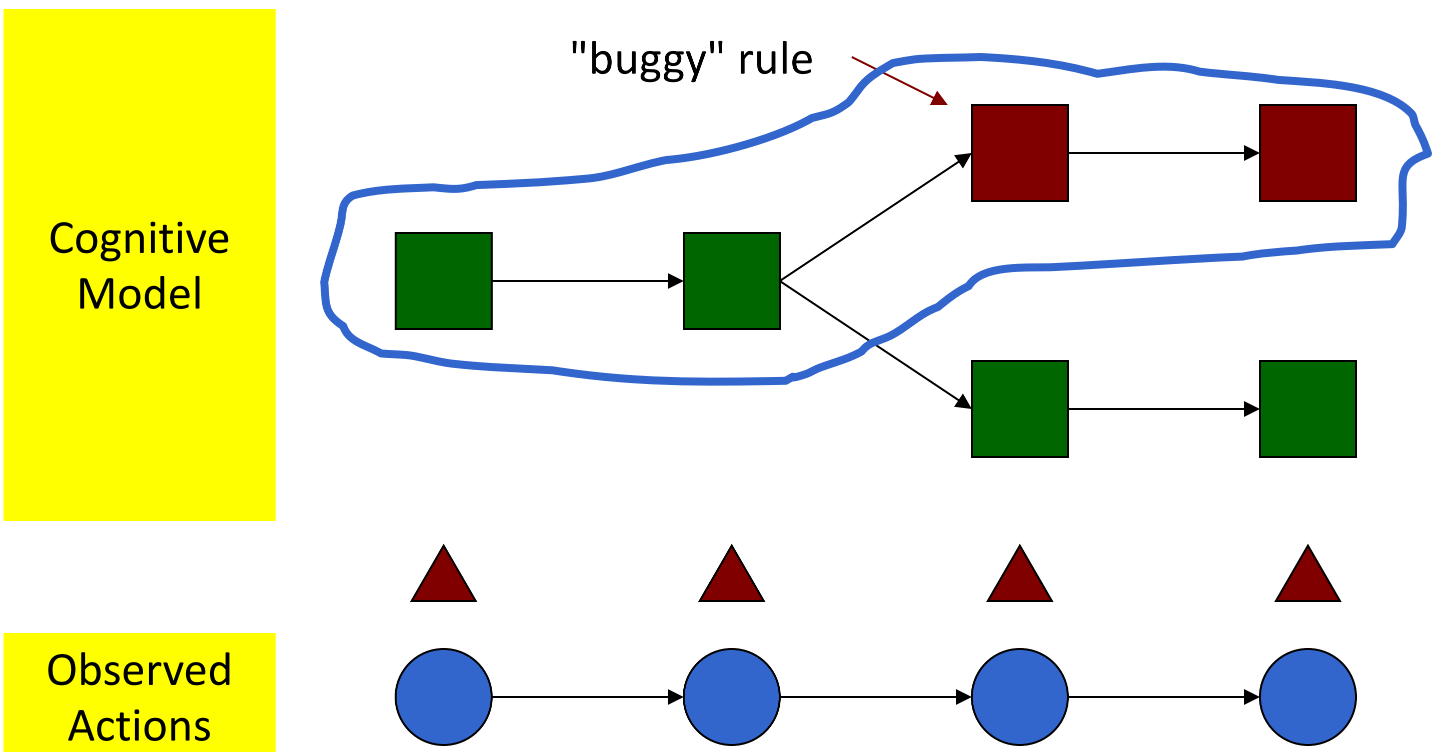


Model tracing

- Major challenges
  - usually assume that one action = one rule firing; what is the general case?
  - what types of "observable actions" should be considered?
  - what about noisy data?
    - e.g., eye-movement data
  - what about "buggy" paths?
    - need to be incorporated!
    - more about this in a bit...

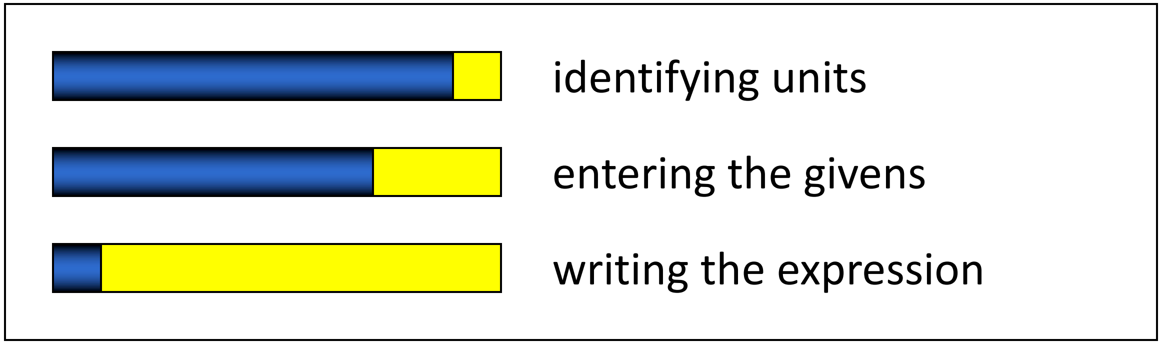
Handling errors

- Include "buggy" rules in our model
- Include help to go with these rules if fired



Knowledge tracing

- Sounds similar, but different from model tracing
- Knowledge tracing = maintaining estimates on how well students know certain skills
  - the "skill-o-meter"
  - example: algebra word problems



Knowledge tracing

- How do we know what students know?
  - use model tracing to determine best matching path through our production system
  - any rules that have fired along this path were used and thus must be known
  - actually, we take into account...
    - probability of doing something right when the rule really isn't known
    - probability of doing something wrong when the rule really is known

Current limitations

- Not a human! Can't (yet) sense emotion, etc.
  - is this always bad though?
- When/how to guide student from passively absorbing knowledge to actively using it
- When/how to correct errors

Take-home message

- Intelligent interfaces use understanding of the current situation to act intelligently
  - this may include user knowledge & skills, current context, adaptation & learning, ...

- Intelligent computer tutors are one good example of an intelligent interface...
  - they utilize a user model of student knowledge
  - they relate user actions to this model to infer current knowledge and goals