

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

$$\begin{bmatrix} x & z \\ 3 & 7 \end{bmatrix}$$

Matrix Multiplication Tutor

Using CTAT with SimStudent and Jess
to create a cognitive tutor

Liz Blankenship
Clearsighted, Inc. and Iowa State University
PSLC Summer School 2007

Task Domain

How can we teach students to multiply matrices?

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \times \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = ?$$

About Me

- B.S. in Computer Science and Math
- Currently employed as a cognitive modeler at ClearSighted, Inc., which has a partnership with Carnegie Learning, enabling cognitive modeling using the Cognitive Tutor SDK

Project Goals

- Learn to use CTAT and associated tools
- Help test SimStudents on a new problem
- Think about cognitive modeling from another perspective

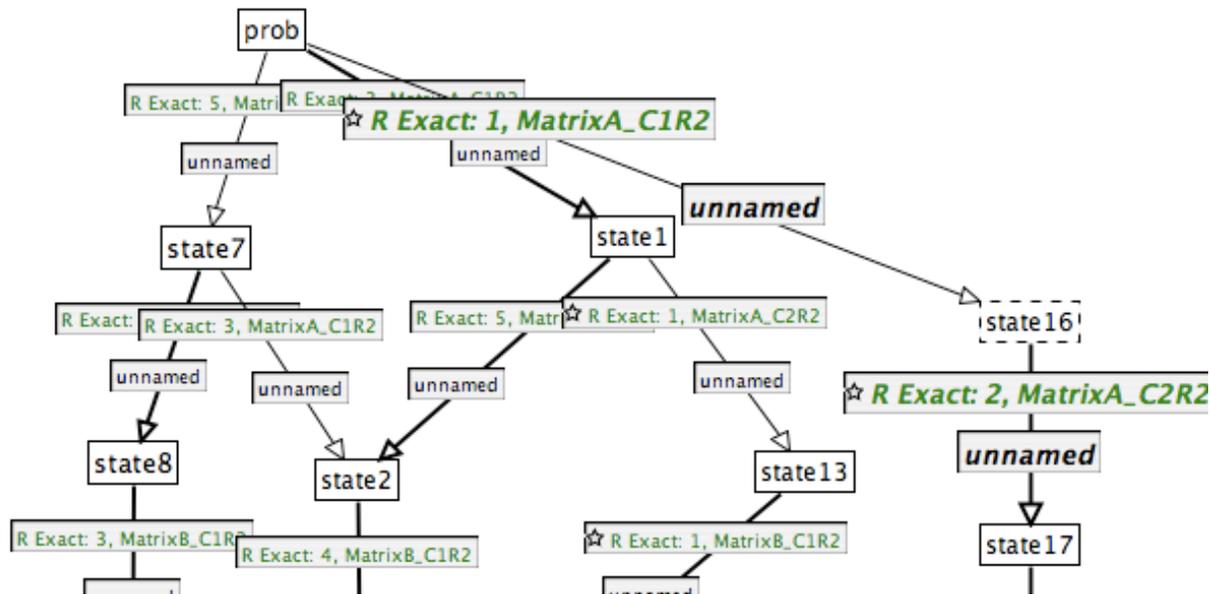
Interface Design

The screenshot shows a Java Swing window titled "Student Interface" with a light blue background. The window contains the following elements:

- Matrix A:** A 2x2 grid with values 1, 2, 3, and 4.
- Matrix B:** A 2x2 grid with values 5, 6, 7, and 8.
- Labels:** "Matrix A", "Matrix B", "A x B unsimplified", "A x B simplified", "Product", and "Final answer".
- Buttons:** "Hint" and "Done".
- Operators:** "x" and "=" symbols are placed between the matrices and their respective simplified and final answer grids.

Using NetBeans,
designing a Java
interface was easy

CTAT for Example-Tracing



CTAT makes creating an example tracing tutor easy, but it is time consuming to create many problems

Cognitive Tutor Design

To create a better tutor that can be reused for multiple problems, a cognitive tutor (or model-tracing tutor) is necessary.

Steps for creating a model-tracing tutor:

- 1) Use Simulated Students (SimStudents) with CTAT to automatically create predicates
- 2) Write any other necessary production rules by hand in JESS

SimStudent

Student Interface

Student

Matrix A

1	2
3	4

Matrix B

3	4
5	6

A x B unsimplified

1*3+2*5	

A x B simplified

Product

Final answer

Hint Done

Using SimStudent, the correct steps to solve the matrix multiplication problem were demonstrated

Auto-generated Rules

SimStudents made rules...

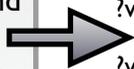
some of them correct...

```
(defrule add-simp
  ?var744 <- (problem (interface-elements ? ? ? ? ? ?var766 ?)
    (matrix-unsimplified nil) (matrix-simplified nil) (matrix-answer nil) )
  ?var766 <- (table (columns $?m783 ?var767 $?) (rows ))
  ?var767 <- (column (cells $?m782 ?var768 $?) )
  ?var768 <- (cell (name ?foa0) (value ?val0&~nil) )

  ?var744 <- (problem (interface-elements ? ? ? ? ? ?var759 ? ?)
    (matrix-unsimplified nil) (matrix-simplified nil) (matrix-answer nil) )
  ?var759 <- (table (columns $?m795 ?var760 $?) (rows ))
  ?var760 <- (column (cells $?m794 ?var761 $?) )
  ?var761 <- (cell (name ?selection) (value ?input&nil) )

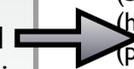
  (test (distinctive ?var768 ?var761))
  (test (same-row ?var768 ?var761))
  (test (same-column ?var768 ?var761))
```

This part correctly identifies a cell (say, with 24+10 in it) and the cell that the simplified equation (34) should be entered into



```
=>
(bind ?input (eval-arithmetic ?val0))
(here-is-the-list-of-foas ?foa0)
(predict-algebra-input ?selection UpdateTable ?input )
(modify ?var761 (value ?*sInput*))
```

Then the correct total (34 in our example) is bound to the correct matrix element



others, not so much...

```
(defrule I-I-unsimplified
  ...
  ;; cut the code for identification of the cells in the row and column

  =>
  (bind ?val4 (add-term ?val2 ?val3))
  (bind ?input (add-term ?val3 ?val4))
  (here-is-the-list-of-foas ?foa0 ?foa1 ?foa2 ?foa3)
  (predict-algebra-input ?selection UpdateTable ?input )
  (modify ?var31 (value ?*sInput*))
)
```

This rule correctly identifies which cells need to be used in multiplication but you can see by the use of **add-term** and no **mult-term** that it inferred the wrong rule.

With more trials SimStudent would generate the correct rules

Problems Encountered

- Running SimStudents required a lot of debugging effort since it had not been thoroughly tested outside of example circumstances
- The learning curve for writing Jess production rules was too steep under the time constraints
- Although SimStudents was on the right track to creating production rules for multiplying matrices, its interface with CTAT prevented me from being able to test the production

Outcomes

- Gained confidence in creation of example-tracing tutors with CTAT
- Gained basic understanding of SimStudents, but it needs more work
- Learned the basics of Jess for implementation of production rules
- Experienced a different perspective of cognitive modeling

Thanks

Thanks to the PSLC and CMU for hosting this summer school, and an extra thanks to Noboru Matsuda, Vincent Alevan, and Jonathan Sewall for many hours of help and interesting discussion in the lab