LEARNING OPERATOR SEMANTICS BY ANALOGY

Sarah A. Douglas
Stanford University and
Xerox Palo Alto Research Center*

Thomas P. Moran
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304

Abstract

This paper proposes a cognitive model for human procedural skill acquisition based on problem solving in problem spaces and the use of analogy for building the representation of operator semantics. Protocol data of computer-naive subjects learning the EMACS text editor suggests that they use their knowledge of typewriting to decide which commands to use in performing editing tasks. We propose a formal method of analysis that compares operators in two problem spaces (based on postcondition similarity) and generates misconceptions (based on pre- and postcondition differences). Comparing these predicted misconceptions with error data and verbal comments in problem solving episodes validates this analysis.

The Phenomena and the Question

Analysis of several experimental protocols of computer-naive people learning the EMACS text editor suggest that they were reasoning by analogy from their knowledge of typewriting. The context of text editing spontaneously evokes the analogy to typewriting, because of the similarity of the keyboards, the similarity of the computer screen to the typed page, and the similarity of the tasks in editing and typing. The use of the typewriter analogy was also prompted by the teachers, with remarks such as: "Yeah, go ahead. It works just like a regular typewriter." The learners' verbal data suggest that they were indeed taking this advice. For example:

> The task is to move to the beginning of a word. The teacher says: "Hitting the space bar still doesn't move you ahead one space." Learner: "That's part of my problem, because on a regular typewriter, you can just zip right in there and do your thing."

Such verbal comments suggest that the learners were engaged in problem solving and were using the analogy to the typewriter to figure out what editing operations were appropriate.

As scientists, we should be skeptical about taking these verbal statements at face value. The question is whether this apparent use of the typewriter analogy actually plays a *significant role* in the learning and performance. Our strategy to examine this question is: Given a general model of analogical learning consistent with the protocols, we develop a specific analysis of the misconceptions that should arise in trying to import knowledge from the typewriter domain to the editor domain. We then test whether these predicted misconceptions are supported by the learners' performance data.

* Beginning September 1983, address will be: Sarah A. Douglas; Department of Computer Science; University of Oregon; Eugene, OR 97403.

A Model of Learning by Analogy

The learner is trying to acquire the cognitive skill required for expert use of a text editor. Text editing skill can be represented as a *problem space* (Card, Moran, & Newell, 1983, Ch. 11). The initial learning task is to build such a problem space. This is done incrementally, not by some sort of pure induction, but rather by borrowing skills from other related domains, which we also consider to be represented as problem spaces.

Learning begins with the learner putting together a rough problem space for text editing from the teacher's instructions. But, when confronted with editing tasks to do, the learner finds that the rough problem space is not yet good enough to support effective problem solving.

We propose that the hardest aspect of learning the problem spaces associated with computer systems is in understanding the operators (i.e., commands). The *operator semantics*, the detailed specifications of how the operators affect the system's conceptual entities, is intricate in computer systems. Thus, the difficulties with the learner's rough initial problem space of editing is due to incorrect knowledge of operator semantics. What the learner does is borrow operators from the typewriter space and apply them in the editing space, which causes unexpected results. This is the source of learners' *misconceptions* about the editor. These misconceptions appear in the learning data as errors, verbal questions, misunderstandings, and inabilities to perform certain types of tasks.

This view differs from other proposals about analogy in AI (e.g., Brown, 1977; Carbonell, 1982; Van Lehn and Brown, 1980; Winston, 1981) and psychology (e.g., Gentner, 1983; Gick & Holyoak, 1983), which involve holistic mappings between domains. For example, Brown (1977) and Winston (1981) map methods from the known domain as entire plans in the new domain. Our model is essentially a scheme for the piecemeal borrowing of fragments of information from the known domain. The problem space formulation is useful, since it provides semi-independent, borrowable procedural units—the operators—which we propose is where most of the misconceptions arise.

Prediction of Specific Misconceptions

We have formalized problem space representations of the typewriter and of the EMACS text editor (Douglas, 1983); and we have developed a method of analysis for comparing the operators in these representations to predict specific learner misconceptions.

Problem space representation. For the typewriter, the primary entity is a spatial array of cells on a *page*, in which each cell is either BLANK or CONTAINS a character. Text entities (words, sentences, paragraphs, etc.) are delineated by blank space. On the other hand, for the EMACS text editor there are two distinct entities, a display *screen* and a *text string*, an internal, invisible sequence of characters.

The screen is generated from the text string by the editor's DISPLAY operation. In addition to text characters, the text string also contains formatting characters (SPACE-CHAR, RETURN-CHAR, and TAB-CHAR) which control where text characters are displayed. The formatting characters also occupy spatial cells on the screen, but they are invisible. Thus, a blank cell on the screen can be either EMPTY or contain an INVISIBLE character.

While the screen looks like a typed page to the learner, its behavior is very different. This behavior can be described by comparing operators in the two domains. We represent operators by precondition and postcondition state predicates. For example, consider the typewriter operator for typing a new character on the page, ADD ($Char_{new}$):

Preconditions
POINTER-AT (*Cell*)
BLANK (*Cell*)

Postconditions
POINTER-AT (NEXT-RIGHT (*Cell*))
CONTAINS (*Cell*, $Char_{new}$)

The preconditions are that the typewriter POINTER must be AT a BLANK cell. The effect of the operator is that the cell CONTAINS the typed character and the POINTER is moved to the NEXT-RIGHT cell. Next, consider the corresponding EMACS operator for typing a new character, INSERT ($Char_{new}$):

Preconditions
*SEQUENCE ($Char_i$, $Char_k$)
*SELECTED ($Char_k$)
POINTER-AT ($Cell_j$)

Postconditions
*SEQUENCE ($Char_i$, $Char_{new}$, $Char_k$)
*SELECTED ($Char_k$)
POINTER-AT (LOCATION-OF ($Char_k$))
{DISPLAY ($Cell_j$, *TEXT-FOLLOWING($Char_i$))}

(The predicates marked with asterisks refer to the underlying text string and hence are invisible to the user.) The preconditions for the EMACS operator are that the editor's POINTER be positioned AT the cell containing the character following the new character (i.e., an "insert before" operator). There is no precondition for blankness. The effect of the operator is to DISPLAY the new character and to re-DISPLAY all the TEXT-FOLLOWING it (which can have complex effects on the screen).

We have in this manner formulated problem spaces for both the typewriter and for EMACS (see Douglas, 1983, for details).

Comparing similar operators. In the process of trying to understand EMACS operators, the learner borrows typewriter operators that are similar to the EMACS operators. We have developed a method of analysis that uses two criteria of similarity: (1) surface feature similarity, which in this case is whether the operators utilize the same keyboard key, and (2) similar effects (e.g., postcondition match). Figure 1 shows all the similarity links between operators.

For example, consider link M8 between the ADD and INSERT operators (which are described above). The postcondition of ADD is that a cell CONTAINS the new character. The postcondition of INSERT has a DISPLAY operation which produces a set of CONTAINS predications, including one for the new character. Thus, the postconditions for ADD and INSERT match.

Once operators have been matched by similarity, their predicate descriptions can be compared for differences in order to generate a taxonomy of potential misconceptions. As both Halasz and Moran (1982) and Gentner (1983) have pointed out, the harmfulness of using analogy as a teaching device is the inability of novices to distinguish the differences from the similarities, that is, novices tend to overextend similarities, thus causing misconceptions. There are two sources of misconceptions when using one operator for another: (1) unknown preconditions and (2) unknown postconditions. Figure 2 summarizes these predicted misconceptions, grouped by the similarity links in Figure 1. These subtle differences are what makes it difficult to learn by analogy. For example, when the learner borrows the ADD operator for the INSERT operator, the BLANK cell precondition comes with the ADD. Thus, we would expect the learner to try to find or create a blank cell before inserting a character. This particular misconception, assuming the BLANK precondition for inserting, is listed as M8a in Figure 2.

An interesting feature of the similarity comparison between typewriter and EMACS operators in Figure 1 is that operators are grouped into two classes: *locative* (for moving the pointer around) and *mutative* (for changing text entities). The figure shows that three locative operators from the typewriter are similar to mutative text editor operators, which insert formatting characters. Some of the major misconceptions for the learners involve an "ontological shift" from locative to mutative operators.

Validation of Predictions with Empirical Data

Our data is taken from the experiments reported by Roberts and Moran (1983). We use detailed protocol records of the learning sessions from their EMACS learning experiments.

The experimental teaching paradigm is tutorial, with a single teacher and a single learner. Interspersed in the paradigm are several quizzes, during which the learner performed typical editing tasks without the help of the teacher. We have two kinds of protocol data to work with—*verbal data*, which was transcribed from audio tape, and *performance data*, consisting of computer-collected time-stamped keystrokes—from which we can see the course of learning in detail.

Performance error data. Figure 2 presents the frequency of learner errors observed in the performance data for the first two quizzes of the learning sessions. The error data shows that semantic errors are a significant portion of early skill acquisition (75 out of 105 errors observed). It can be seen that the overall coverage by the above analysis is good (covering 62 of 75 semantic errors). Errors involving the "ontological shift" (misconceptions M5 through M7) are very frequent, accounting for 30 of the observed errors.

Verbal protocol example. Another use of these analytically-generated misconceptions is to explain some of the confusions found in verbal protocols, such as the example at the beginning of this paper. In this example, the misconception of inserting an invisible formatting character is evident. The novice uses the space bar to move across the screen without understanding that she is inserting a space *character* into the text string (misconception M5 in Figure 2). For some reason she forgets to use the text editor command (which she knows and has used many times before) to move over one horizontal unit. The effect of inserting invisible characters is very surprising to her, and she cannot explain why the cursor maintains the same distance from her target location while moving further to the right across the screen. She is frustrated in her attempts to "just zip right in there." This example also illustrates how the interactive nature of the text editor reduces the credit/blame assignment problem, since the novice can see the immediate effects of applying the wrong operator and can localize where to correct the problem.

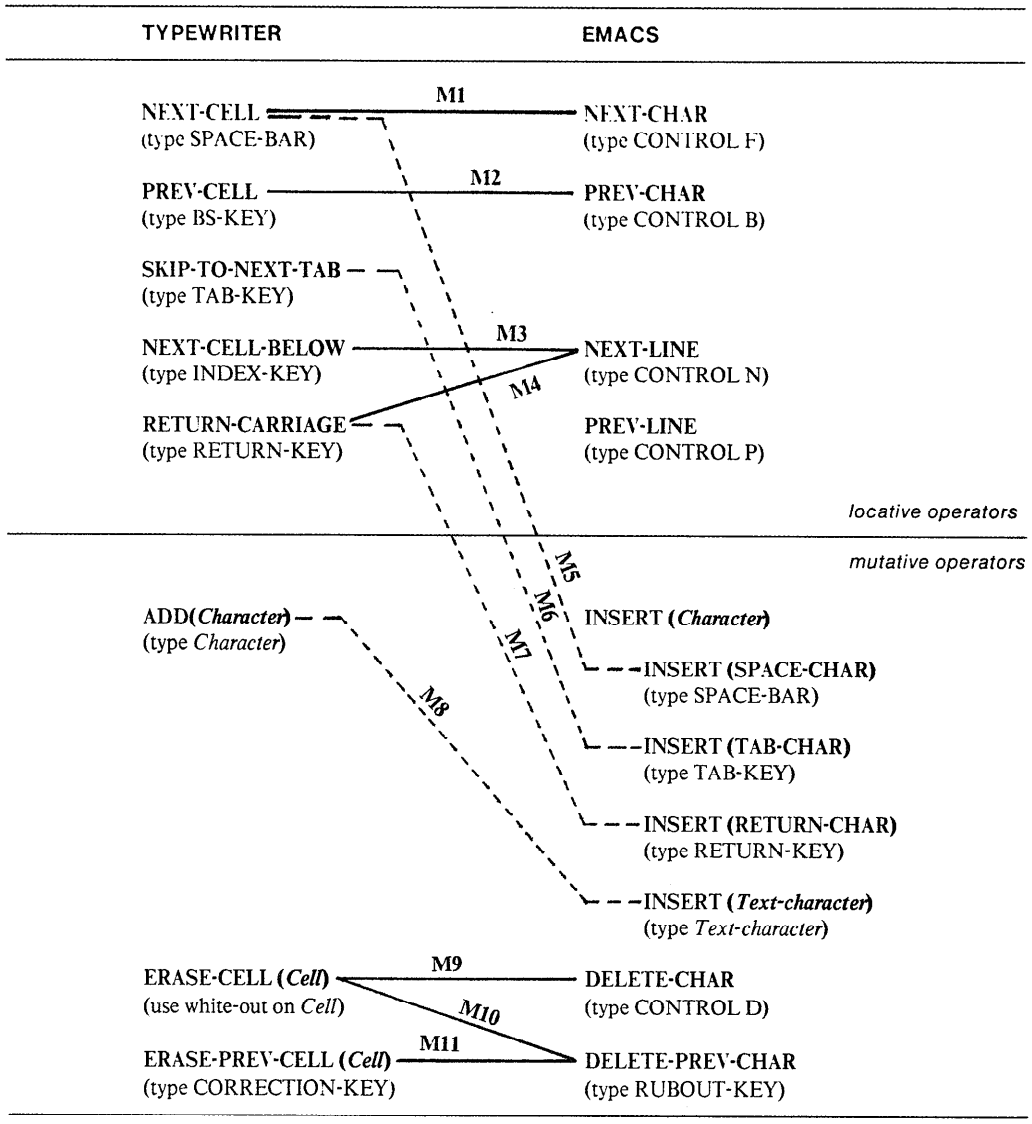

Figure 1
Similarity Links between Typewriter and EMACS Operators
(Solid lines indicate similar-action relations; and dashed lines indicate similar-action and same-key relations.)

Conclusions

Our initial observation, that text editor novices rely heavily on their knowledge of typewriting to understand the semantics of text editor operators, accounts for learners' performance errors resulting from misconceptions and for many confusions found in the verbal protocols. We have presented an analysis technique based on problem spaces for generating a taxonomy of analogical misconceptions. The analysis identifies analogous operators by similarity of postconditions (effects) and predicts misconceptions by overextending the similarities to other operator postconditions and to operator preconditions.

The use of analogy for teaching has both advantages and disadvantages. Individual situations must be analyzed to determine whether the net balance of using an analogy is for the good. In this paper we have attempted to develop an analysis technique to predict the specific effects of using a specific analogy. This operator mapping technique is a potentially useful design tool for elaborating the conceptual complexity resulting from the interaction of new knowledge of a system to be learned with previous knowledge of other systems.

	Predicted Misconceptions	Frequency of Observed Errors
M1.	NEXT-CELL ---- NEXT-CHAR	
	a. Ignorance that NEXT-CHAR moves by characters, not cells of text string.	6
M2.	PREV-CELL ---- PREV-CHAR	
	a. Ignorance that PREV-CHAR moves by characters, not cells.	—
M3.	NEXT-CELL-BELOW ---- NEXT-LINE	
	a. Ignorance that NEXT-LINE moves by characters and columns, not cells.	—
M4.	RETURN-CARRIAGE ---- NEXT-LINE	
	a. Ignorance that NEXT-LINE moves by characters and columns, not cells.	1
M5.	NEXT-CELL ---- INSERT(SPACE-CHAR)	
	a. Ignorance that cell contains an invisible SPACE-CHAR.	7
M6.	SKIP-TO-NEXT-TAB ---- INSERT(TAB-CHAR)	
	a. Ignorance that TAB-CHAR is an invisible character, not a cell.	4
M7.	RETURN-CARRIAGE ---- INSERT(RETURN-CHAR)	
	a. Ignorance that RETURN-CHAR is an invisible character, not a cell.	19
M8.	ADD(*Character*) ---- INSERT(*Text-character*)	
	a. Assumes typewriter precondition BLANK(*Cell*).	9
	b. Ignorance that postcondition inserts character into string, not cell (i.e. no strikeover).	2
	c. Ignorance that postcondition inserts character in front of currently selected character.	4
	d. Ignorance that postcondition re-displays text to the right of the insertion point.	—
M9.	ERASE-CELL ---- DELETE-CHAR	
	a. Assumes typewriter precondition CONTAINS(*Cell, Character*).	—
	b. Assumes typewriter postcondition BLANK(*Cell*) instead of re-displaying text string.	3
M10.	ERASE-CELL ---- DELETE-PREV-CHAR	
	a. Assumes typewriter precondition CONTAINS(*Cell, Character*).	—
	b. Assumes typewriter postcondition BLANK(*Cell*).	—
	c. Ignorance that postcondition deletes character in front of currently selected character.	7
M11.	ERASE-PREV-CELL ---- DELETE-PREV-CHAR	
	a. Assumes typewriter precondition CONTAINS(*Cell, Character*).	—
	b. Assumes typewriter postcondition BLANK(*Cell*).	—
	Observed errors covered by the predicted misconceptions:	**62**
	Other observed semantic errors not covered by the predicted misconceptions:	**13**
	Observed syntactic errors (not covered by the predicted misconceptions):	**28**
	Observed typing errors (not covered by the predicted misconceptions):	**2**

Figure 2
Predicted Misconceptions and Frequency of Observed Performance Errors for Four EMACS Learners

References

Brown, R. *Use of analogy to achieve new expertise* (AI-TR-403). Boston, MA: MIT, Artificial Intelligence Lab, 1977.

Carbonell, J. Learning by analogy: Formulating and generalizing plans from past experience. In R.S. Michalski, J.G. Carbonell, & T.M. Mitchell (Eds.), *Machine Learning*. Palo Alto, CA: Tioga Publishing Co., 1982.

Card, S. K., Moran, T. P., & Newell, A. *The psychology of human-computer interaction*. Hillsdale, N.J.: Erlbaum, (1983).

Douglas, S. A. *Learning to text edit: Semantics in procedural skill acquisition*. Ph.D. dissertation, Stanford University, 1983.

Gentner, D. The structure of analogical models in science. In D. Gentner and A. S. Stevens (Eds.) *Mental models*, Hillsdale, N.J.: Erlbaum, 1983.

Gick, M.L. & Holyoak, K.J. Schema induction and analogical transfer. *Cognitive Psychology*, 1983, *15*, 1-38.

Halasz, F., & Moran, T. P. Analogy considered harmful. *Proceedings of the Human Factors in Computer Systems Conference*, Gaithersburg, MD. March 15-17, 1982.

Roberts, T. R., & Moran, T. P. Learning and reasoning by analogy. *Communications of the ACM*, 1983, *25*, 265-283.

VanLehn, K., & Brown, J. S. Planning nets: A representation for formalizing analogies and semantic models of procedural skills. In R.E. Snow, P. Federico, & W.E. Montague (Eds.), *Aptitude, learning, and instruction, Vol. 1*. Hillsdale, N.J.: Erlbaum, 1980.

Winston, P. Learning and reasoning by analogy. *Communications of the ACM*, 1981, *23*, 689-703.