Towards Real-Time GOMS

Bonnie E. John[1] Alonso H. Vera[2] Allen Newell

28 December 1990
CMU-CS-90-195 2

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA, 15213-3890

Abstract

We present an analysis of an expert performing a highly interactive computer task. The analysis uses GOMS models, specifying the *G*oals, *O*perators, *M*ethods, and *S*election rules used by the expert; the GOMS models are implemented within an unified theory of cognition called Soar. Two models are presented, one with *function-level* operators which perform high-level functions in the domain, and one with *keystroke-level* operators which describe hand movements. For a segment of behavior in which the expert accomplished about 30 functions in about 30 seconds, the function-level model predicted the observed behavior well, while the keystroke-level model predicted only about half of the observed hand movements. These results, including the discrepancy between the models, are discussed.

This research was supported by the Office of Naval Research, Cognitive Science Program, Contract Number N00014-89-J-1975N158. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Office of Naval Research or the U. S. Government.

[1] School of Computer Science and Department of Psychology

[2] Department of Psychology

Keywords: user models, cognitive models, GOMS, Model Human Processor, Soar

MOTIVATION

This research was done in preparation for a panel at the Human Factors Society (HFS) Meeting in Orlando, Oct. 5-12, 1990, organized by Wayne Gray and Mike Atwood of the Intelligent Interfaces Group at the NYNEX Science and Technology Center. The purpose of the panel was to demonstrate GOMS analyses to the human factors (HF) community in an exciting manner so that HF practitioners would be motivated to learn more about GOMS and use it in their work.

Three applied psychology researchers (Judy Olson of the University of Michigan, Jay Elkerton, of Philips Laboratories, and Bonnie John of Carnegie Mellon University) agreed to analyze a task selected by Wayne Gray. Gray selected a human-computer interaction domain, made a videotape of an expert interacting in that domain, and gave the videotape and an associated transcription to each researcher (Appendix I). In the interest of enticing HFS conference participants to the panel, and stretching the GOMS methodology to include real-time, interactive domains, Gray selected a domain hitherto unanalyzed with GOMS: video game play (in particular, Nintendo's Super Mario Bros. 3[1] adventure game). The four panelists (the three previously named researchers and Wayne Gray, himself) each did a GOMS analysis of the interaction independently, and presented the results to each other, and to the HFS audience, for the first time during the panel session. Because of limited presentation time, the panelists restricted their analyses to the first 27 seconds of the expert's interaction (a natural break in the play, where the expert flies the game's main character, Mario, off the screen and to a different part of the game's world). The panel continued with a discussion of the similarities, differences, and uses of the analyses. Informal comments to the panelists indicate that the attendees were indeed excited by the presentations and were motivated to learn more about GOMS, with the hope of using it in their work.

Our own research interests lie in the creation of *engineering models* of computer users that allow quantitative prediction of hard measures (performance time, learning time, the commission, detection , and repair of errors, etc.) in the specification stage of system design (see Newell & Card, 1985, 1986; John, 1988 for detailed discussions of this position). We are currently extending GOMS to highly interactive task domains. By highly interactive tasks, we mean tasks in which a user perceives a display, comprehends and responds to that display on the order of once every second, for several seconds at a time. We call this rapid interaction the *immediate interaction cycle*. Tasks displaying the immediate interaction cycle are common in human-computer interaction (HCI): searching for information with computer browsers, constructing a new diagram with an interactive graphics package, playing "what-if" with a spreadsheet, creating a schedule with PERT charts, exploring mathematical relationships with graphical statistics programs, creating real-time, multi-media presentations. The list seems endless. Video games are extreme examples of such tasks. The interaction in many video games seems almost manically driven by the game rather than by the player. Goals seem to be interrupted, suspended and returned to. Yet, similar to other domains successfully modelled by GOMS, an expert's behavior looks to be highly knowledge intensive and eventually seems to become routine. We welcomed the opportunity to stretch our analysis techniques to the challenge of Super Mario Bros. 3.

[1] Super Mario Bros. 3, Nintendo, and the games and characters discussed in this paper are trademarks of, and under copyright to, Nintendo of America, Inc.

This technical report provides documentation of the analyses presented by Bonnie John at that HFS panel. The content of the analyses remains essentially the same. At the time of the presentation, many assumptions and decisions made were made from intuitions born of intimate knowledge of GOMS analyses. Here, we present the logic underlying those initial intuitions. We hope this report will further encourage the panel's attendees, and others in the fields of applied psychology and cognitive modeling, to learn more about the GOMS methodology, and the extensions to it made with the Soar unified theory of cognition described herein.

GOMS MODELS

GOMS is a formalism for representing routine cognitive skill. The original concept appeared in *The Psychology of Human-Computer Interaction*, by Card, Moran, and Newell (1983, and also in an earlier article, Card, Moran, & Newell, 1980) as a model of the performance of computer users. GOMS stands for *G*oals, *O*perators, *M*ethods, and *S*election rules. A *goal* is a symbolic structure that defines a state to be achieved, and determines a set of methods by which it may be accomplished. *Operators* are elementary perceptual, cognitive, or motor acts, whose execution is necessary to change any aspect of the user's mental state or to affect the task environment. An operator is defined by a specific effect (output) and by a specific duration. An operator may take inputs, and its outputs and duration may be a function of its inputs. The selection of operators for any specific GOMS model defines its grain of analysis.[2] *Methods* are procedures for accomplishing a goal. A method is a sequence of goals and operators, with conditional tests on the contents of the user's immediate memory and on the state of the task environment. In routine cognitive tasks, methods are assured of success (up to the possibility of having been mis-selected, errors in implementation, and the reliability of the equipment). By contrast, in problem-solving tasks, methods may or may not lead to success depending on the user's lack of knowledge or understanding of the task environment. Also, GOMS methods are procedures that the user already has at performance time, as opposed to plans created during task performance. If more than one method can be used to accomplish a goal, a selection must be made. The essence of skilled behavior is that these selections are not problematic, that they proceed quickly and smoothly. Although the original description of GOMS acknowledges that extended (but still unproblematic) decision processes may be necessary in some situations, *selection rules*, which are if-then rules that recognize the method appropriate for the specific task situation, are a signature of routine cognitive skill and GOMS.

This original specification of GOMS had many limitations. Among them, GOMS predicted only error-free behavior with no mechanisms for predicting the occurrence of errors, neither frequency, type, nor when they might occur. There was no mechanism for learning. The goal-stack control structure was inadequate for handling interruptions. Operators in the original GOMS model were on the order of one half second, at their smallest grain of analysis, and many contained mixtures of perceptual, cognitive and motor acts. This confounding of processes dictated that operators be sequential, whereas many skilled tasks exhibit concurrent perceptual, cognitive, and motor acts.

Although not limited by the architecture of GOMS, the example tasks in the 1983 volume implicitly defined boundaries of GOMS analyses. These early tasks had a static visual

[2] Card et. al. (1983, Chapter 5) demonstrate GOMS models of text-editing at four levels of analysis, from the unit task level where each editing modification (e.g., delete a paragraph) is accomplished in a single EDIT-UNIT-TASK operator, down to the keystroke level where the operators were keystrokes and mouse-movements.

display for the inputting information to the user (e.g., marked-up manuscript for text editing, sketch of a circuit for VLSI layout). They allowed the user to work at her or his own pace; the user did not have to wait for critical information to appear, and critical information did not disappear. Subtasks were not thrust upon the user by a changing environment. Thus, these early studies did not demonstrate the appropriateness of the GOMS models for interactive tasks.

Since its introduction, GOMS analyses have been used to model many tasks in a diversity of domains and several extensions have been made to the original formulation (Olson & Olson, 1990). GOMS models have been demonstrated within a production system architecture to model performance and learning (Kieras & Polson, 1985; Polson & Kieras, 1985; Kieras & Bovair, 1986; Singley & Anderson, 1989). The operators have been brought down to the level of elementary perceptual, cognitive, and motor operations (Rosenbloom, 1983; John, Rosenbloom, & Newell, 1985). This allows perceptual, cognitive, and motor processes to be expressed as occurring in parallel when the task environment allows look-ahead or anticipation of operations (John & Newell, 1989). Errors attributed to working-memory limitations have been predicted (Lerch, Mantei, & Olson, 1989). Verbal input and output and tasks driven by the environment have extended the characteristics of GOMS tasks (John, 1990). Task domains modelled in GOMS now include graphic editors (Ziegler, Hoppe, & Fahnrich, 1986), spreadsheets (Olson & Nilsen, 1988), computer command abbreviations (John, et al., 1985; John & Newell, 1987) oscilloscopes (Lee, Polson, & Bailey, 1989), touch typing (John, 1988; John & Newell, 1989), and telephone operator call handling (John, 1990; Gray, John, Stuart, Lawrence, & Atwood, 1990).

GOMS, THE MODEL HUMAN PROCESSOR, AND SOAR

Since GOMS provides a formalism within which to describe and predict a range of user behaviors, it could be considered a closed model, sufficient in itself. In actuality, GOMS presupposes an underlying general theory of human cognition, for which it is the specialization and instantiation of that theory to specific task environments and types of humans. GOMS was initially presented exactly this way (Card, et. al., 1983). The underlying cognitive theory was the Model Human Processor (MHP). It comprised an architectural structure, composed of memories (long-term memory, working memory and sensory buffers) and processors (perceptual, cognitive and motor). The operation of this structure was not given in full detail (as in a computer architecture). Instead a series of operating principles were given, such as decreasing speed of the cognitive processor with uncertainty, and the use of problem spaces as an overall way of organizing performance. The MHP was meant to summarize, circa the early 1980's, what had been learned in cognitive science about the operation of basic information processing. The version of GOMS originally presented was entirely consistent with the MHP. In turn, the MHP was meant to do more than simply justify GOMS. It was to provide a basis for reasoning more generally about users interacting with computers. (Indeed, it made no pretense to be a general model of all human cognition, but was itself a specialization to the situations of interest to HCI.)

The MHP was far from a complete cognitive theory, even in the HCI domain to which it was specialized. The state of the art of cognitive science was simply not sufficient to provide such a theory. Nevertheless, the MHP has proved sufficient to conceptually support all of the extensions that have been made to GOMS. The MHP was cast in terms of productions and recognition-act cycles; this provided an alternative formalism to the procedural-language formalism of the original GOMS, permitting the extension to skill learning. The MHP defined enough internal memory structure for the cognitive processor to support the refinement of GOMS operators from the half second level down to the 50 msec level. This memory structure was also sufficient to support the extension of GOMS to account for some working memory errors. And, finally, the three-processor structure of the MHP (perceptual, cognitive and motor) was sufficient to extend GOMS to continuous interaction tasks, such as typing and talking-while-keying tasks, where the human operates in a pipelined, overlapped mode.

Although it is impressive that the MHP has been conceptually rich enough to support all these extensions, the MHP remains a highly sketchy theory. The memories are characterized only by a few descriptive parameters (such as half-life) and the principles of operation are not computationally described (for example, it is simply stated that the power law of practice is obeyed). It is not literally possible to derive GOMS (in any of its variants) from the MHP. Additional detail is always required, for example, the particular procedural language structure of the original GOMS. What remains true is that detailed GOMS systems are further specifications of the structure and principles of the MHP. Finally, unlike GOMS itself, which has been continuously tested and extended since it was initially introduced, no such activity of updating, exercising or refinement has occurred for the MHP. Thus, good reasons exist to adopt a better base for GOMS analyses than the MHP, if a suitable one can be found. There are some minimal conditions for any basic psychological theory to support GOMS, namely, that it support the particular control structure of goals, operators, methods and selections rules. Many cognitive architectures can do this, but certainly not all, e.g., Act* (Anderson, 1983) can, but in general connectionist architectures cannot (Rumelhart, et. al., 1986). Furthermore, the real leverage in a new base should be the ability to support further extensions of GOMS and its integration with the many other cognitive and perceptual processes that are relevant to interacting with computers.

Soar is a recent attempt to provide an architecture for human cognition (Laird, Newell & Rosenbloom, 1987; Newell,1990). It is generally consonant with the MHP, and hence supports the basic control mechanisms of GOMS. But, unlike the MHP, Soar is a completely specified architecture. That is, rather than being given as an abstract memory/process structure plus an set of abstract principles of operation, Soar is given in the fashion of a programmed computer, with data structures, memory accessing organization, and full details of the operation of the processors. Thus, one can specify the contents of Soar memory structures for particular users in particular task situations. These contents, in effect, program Soar so that it produces simulations of the behavior of the user in the task.

As succinctly described in (Lewis, et al., 1990) and in more detail elsewhere (Laird, et. al., 1987; Newell,1990), the Soar architecture formulates all tasks in *problem spaces*, in which *operators* are selectively applied to the *current state* to attain *desired states*. Problem solving proceeds in a sequence of *decision cycles* that select problem spaces, states and operators, resulting in the application of the operator to move to a new state in the space. Each decision cycle accumulates knowledge from a long-term *recognition memory* (realized as a production system). This memory continually matches against *working memory*, elaborating the current state and retrieving preferences that encode knowledge about the next step to take. Access of recognition memory is involuntary, parallel, and rapid (on the order of 10 msec). The decision cycle accesses recognition memory repeatedly until quiescence, when no more knowledge can be brought to bear; then the decision is made about which step to take next. Each decision cycle takes on the order of 100 msec.

If, at quiescence, the accumulated coded knowledge in working memory is insufficient (or conflicting), so that Soar's next step cannot be determined, then an *impasse* occurs. Soar responds to an impasse by creating a subgoal in which a new problem space can be used to acquire the needed knowledge (or resolve the conflict). If, similarly, lack of knowledge prevents progress in this new space, another impasse occurs and another subgoal is created -- and so on, leading to an entire goal-subgoal hierarchy. Once an impasse is resolved by problem solving, the *chunking* mechanism adds new productions to the recognition memory that encode the results of the problem solving. Thus, the impasse is avoided in the future, because these productions provide the appropriate knowledge immediately.

Soar interacts with the external environment through perceptual and motor processes, which operate through the working memory. Incoming perceptions are added to the current state in the top problem space and motor commands are made part of this current state. The interactions occur asynchronously with the operation of the cognitive decision cycle.

Soar basically subsumes the MHP and provides a much better basis on which to construct GOMS models. However, the situation is not perfect. It remains an open issue to show that some of the MHP operating principles expressing global psychological laws (such as Fitts' law and Hick's law) follow from the Soar architecture. (Some global laws, such as the power law of practice, have already been shown to be characteristic of the architecture.) Some of the ways in which Soar does not yet adequately subsume the MHP relate to the perceptual and motor processors, which remain underspecified aspects of the Soar architecture. These difficulties do not substantially affect taking GOMS models as specializations of the Soar architecture for appropriate users doing appropriate tasks. In fact, the basic GOMS model refined for 50 msec operators has been derived from Soar (Newell, 1990, Chapter 5). This extends easily to longer duration GOMS operators, to the learning effects (in terms of Soar chunking), and to the three-processor GOMS models of continuous behavior, although all the details have not yet been fully worked out.

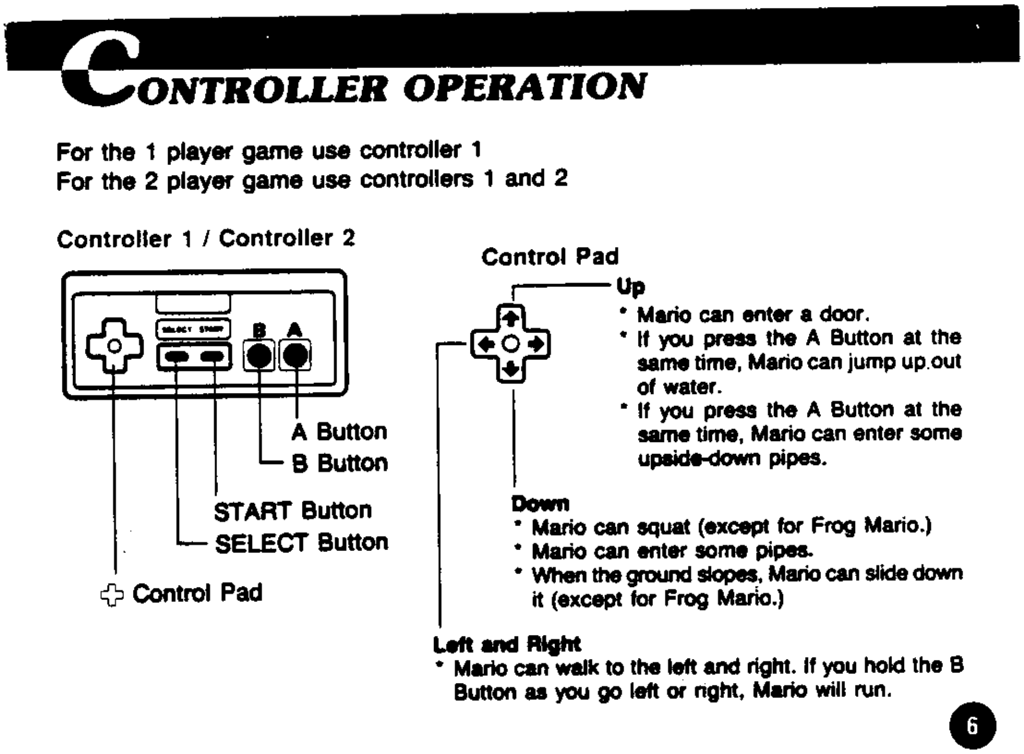
Soar is a better basis than the MHP on which to construct GOMS models for reasons that go beyond just reproducing the current GOMS, even if that can be done more elegantly. The most important is that Soar is complete enough and cognitively adequate enough so it is being used to model and explain many diverse cognitive phenomena (Lewis, et. al. 1990; Newell, 1990). Examples include natural language comprehension, problem solving, immediate reasoning, perceptual search, strategy discovery and change, and the taking of instructions. These tasks encompass the major dependent variables of interest in HCI -- time, errors, and learning rates. Thus GOMS automatically is subsumed within a theory which is being extended to many other areas of cognition relevant to HCI. These extensions, even when good psychology, do not always automatically extend GOMS in ways that preserve its property of permitting engineering calculations and making parameter-free predictions. But they provide excellent starting points for such extensions. The second reason is the universal availability of simulations using Soar, so that even when analytic calculations cannot be made, it is possible to put all the various parts of a user model together and simulate what should happen (John, Newell & Card, 1990, is an example in the domain of user's behavior with a browsing system).

Thus, from now on in this paper we will work within the Soar architecture, taking it as the architectural foundation upon which to build GOMS models.

THE TASK

Nintendo's Super Mario Bros. 3 was chosen for the HFS panel by Wayne Gray because it is a highly interactive task domain, its popularity as a video game was likely to draw a large audience to the panel, and an expert, KP, was readily available and willing to participate.

Super Mario Bros. 3 is a typical adventure game with treasure to collect, enemies to avoid or kill, and super powers to acquire and use. The user manipulates the hero, Mario, through the world by pressing buttons on a hand-held controller (Figure 1). The game has eight different worlds to traverse, and each world has several levels, with the difficulty of play increasing with the world and level number.



CONTROLLER OPERATION

For the 1 player game use controller 1
For the 2 player game use controllers 1 and 2

Controller 1 / Controller 2

A Button
B Button
START Button
SELECT Button
✣ Control Pad

Control Pad

Up
- Mario can enter a door.
- If you press the A Button at the same time, Mario can jump up out of water.
- If you press the A Button at the same time, Mario can enter some upside-down pipes.

Down
- Mario can squat (except for Frog Mario.)
- Mario can enter some pipes.
- When the ground slopes, Mario can slide down it (except for Frog Mario.)

Left and Right
- Mario can walk to the left and right. If you hold the B Button as you go left or right, Mario will run.

6

Figure 1. The hand-held controller for Super Mario Bros. 3. (from p. 6, *Super Mario Bros. 3. instruction booklet*, reprinted by permission of Nintendo of America, Inc.)

Several enemies populate these worlds, among them are *Goombas*, who can kill Mario by running into him but can be killed by Mario, *ParaGoombas* (flying Goombas with wings), *Venus Fire Traps*, invincible plants who live in pipes and throw deadly *fireballs*, and *Koopa Troopers (Koopas)*, turtle-like creatures who retreat into their shells, which can then be kicked or thrown by Mario to defeat other enemies or break open treasure blocks.

Treasures in these worlds include coins that allow Mario to buy additional lives, and mushrooms and leaves that give Mario super powers when he runs into them. Treasures can appear directly on the screen, or they can be hidden in blocks and only appear when the blocks are broken by Mario. Both killing enemies and collecting treasures give Mario points.

The display that appears to the user at the beginning of each game is shown in Figure 2. This display shows Mario, four question blocks, a type of block guaranteed to contain a treasure (QB.1 through QB.4), a Goomba (G.1), the level ground that Mario walks on (Ground1.1), and a scaffold he can climb on to reach treasures or avoid enemies (Scaff2.1).

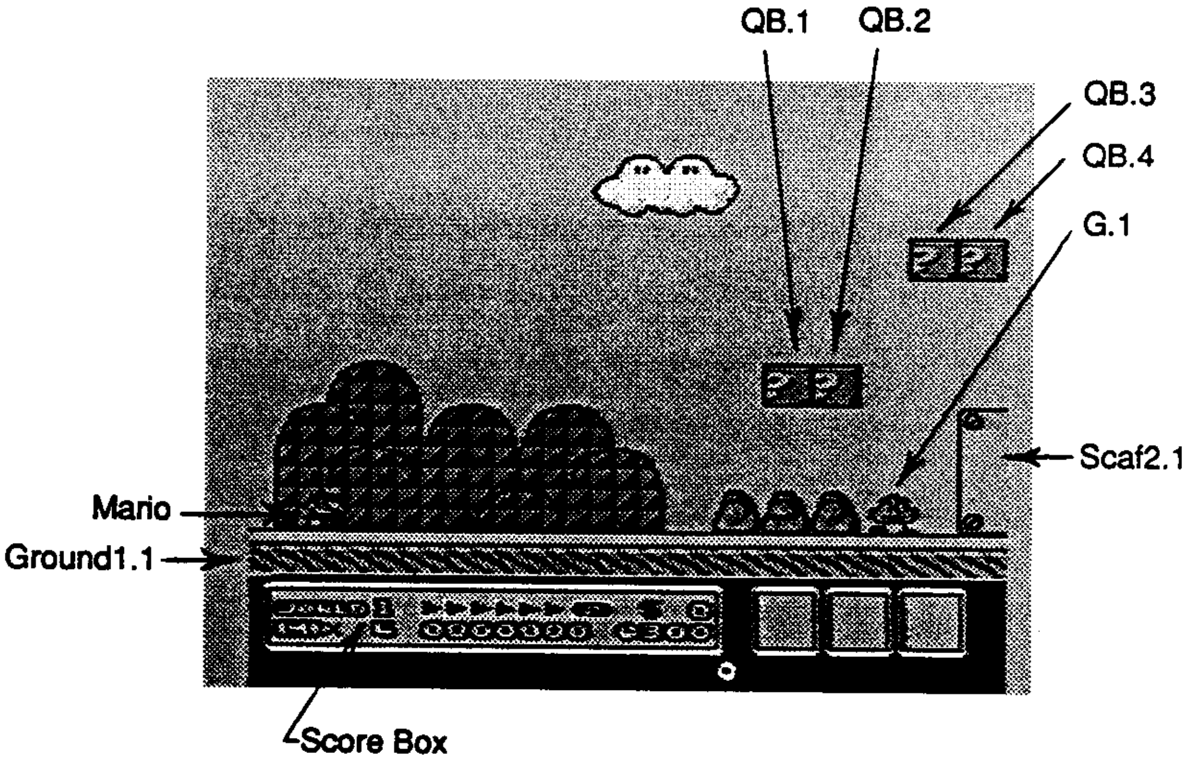


Figure 2. Start-up display for Super Mario Bros. 3 World 1 Level 1.

For simplicity sake (for both the Nintendo-naive panelists and the ease of generating data) Gray chose the task to be the lowest level of difficulty, World 1 Level 1. He asked the nine-year-old KP to traverse World 1 Level 1, collecting as many points as possible, thinking aloud as he played. Gray videotaped KP performing this task to provide observed behavior against which to measure the predictions of the GOMS analyses. The panelists were asked to analyze the first 27 seconds of play, in which time KP collected nine treasures, killed five enemies, avoided three dangers, added two super powers, and flew off the screen to another part of World 1 Level 1. The completion of World 1 Level 1 took KP and additional 43 seconds. Appendix I contains the transcription of KP's behavior for the analyzed segment.

TWO GOMS ANALYSES

GOMS suggests that the goals, operators, and methods can be defined from an objective analysis of the task. Optimal selection rules can often also be specified from a task analysis, but experience with human behavior indicates that selection rules in actual use are specific to an individual user and must be inferred from his or her behavior. We conducted the GOMS analyses with as little reference to the expert's behavior as possible, opting instead, for taking the knowledge explicit in the instruction booklet and reasoning about the task itself. We found that the goals, operators, and methods sufficient to play the game could be inferred from the instruction booklet and task analysis. Selection rules were determined through reference to the expert's behavior.

The analyses were carried out at two levels: the *function-level* where operators are at the level of gathering an item that gives points or killing an enemy, and the *keystroke-level*, where operators are at the level of individual finger movements on the game's control panel. Each level is considered a separate GOMS analysis, as in Chapter 5 of *The Psychology of Human-Computer Interaction* (Card, et. al. 1983) where nine models at four different levels were compared. We will present these analyses in parallel, first describing the goals, operators, methods and selection rules for each, then describing the process of applying this information to the task of playing Super Mario Bros. 3.

Goals, operators, methods, and selection rules of the task domain

The instruction booklet is the primary source of knowledge for this GOMS analysis. It provides the overall *goals* of the game: clear the level (in order to go on to the next level), increase your standing by accumulating points, coins, cards, and extra lives (these are the things measured in the score box at the bottom of the screen), and avoid death due to being touched by an enemy, falling into a hole or fire, or running out of time.

These goals can be accomplished by performing *function-level operators* (FLOs), also inferred from the instruction booklet. Figure 3 lists the goals of the game, the FLOs that act to fulfill those goals in the segment of behavior analyzed, and the passages in the instruction booklet from which the FLOs were inferred. In service of clearing the level, the FLOs are move-towards-end (in effect, moving to the right) and touch-goal, where the goal is a white box in the middle of a black screen at the end of the level (Mario must jump up to touch it). To increase standing, the FLOs are gather-item, search-in-block, and attack-enemy. Gather-item collects items on the screen that give points or money. Search-in-block hits blocks to see if something valuable comes out, which could then be gathered. Points can also be gotten with the attack-enemy FLO, although there is some risk involved; an attack could fail and the enemy could kill Mario, violating the goal to avoid death. To fulfill the goal to avoid death, the FLO is avoid-danger, where a danger can be an enemy, a hole in the ground (if Mario falls in, he dies), or fire. There is no FLO that prevents Mario from running out of time. However, a consideration of time is inherent in the selection among the other FLOs.

FLOs are realized through *keystroke-level operators* (KLOs).[3] The KLOs are the hand movements necessary to manipulate Mario, and involve hitting the six buttons on the controller (Figure 1). The four buttons in a cross configuration move Mario to the right, left, up, and down[4] and we call the KLOs press-right, press-left, press-up, and press-down, respectively. The A-button makes Mario jump, and when he has a tail, pressing the A-button repeatedly makes Mario fly. Holding the B-button down allows Mario to pick up things that lie next to him, and to accelerate when used in conjunction with the right- or left-buttons. Releasing the B-button causes Mario to drop whatever he is carrying.

These two levels of operators have a relationship described by Card, et al. (1983, Chapter 5) as being formed by *splitting operators*. That is, the FLOs are *split* into the hand

[3] I use the terms *goals, function-level operators* and *keystroke-level operators* to distinguish between three different levels in the typical GOMS goal hierarchy. This distinction is drawn to make the different levels easy to write about; it does not reflect a theoretical distinction between goals and operators. As Kieras (1988) observes for GOMS analyses in general, "this distinction is intuitively-based, and it is also relative; it depends on the level of analysis."

[4] Note that pressing the up button does not make Mario jump up, rather it allows him to move up in certain situations: to go up a pipe with its opening over his head (together with the A-button) or pop out of water (together with the A button) or to go through a door. None of these maneuvers are necessary in segment of behavior analyzed here. Likewise, the down button makes Mario squat, go down a pipe when he is standing on its opening, or to slide down a hill. Again, none of these maneuvers are used in this segment of behavior.

GOAL: FUNCTION-LEVEL OPERATOR	PARAMETERS	INSTRUCTION BOOKLET SOURCE	PAGE
Clear-level:		"...touch the goal to...clear the level"	19
Move-towards-end		"At the end of each action scene..."	19
Touch-goal		"...touch the goal to...clear the level"	19
Increase-standing:	no.of Marios remaining score,number of coins, cards taken	Score box at bottom of screen	16
Search-in-block	Question blocks, other blocks	"Hit blocks...A useful item might pop out!"	11
Gather-item	Coin, Starman, Super Leaf, 1-Up Mushroom, Super Mushroom, Fire Flower	"Gain more power by gathering items"	18
Attack-enemy		Points are given for attacking enemies	14
	Goomba	Attacks pictured	12,14
	Koopa	"After you have jumped on a Koopa..."	8
	Para-Goomba	"Once you jump on it..."	35
Avoid-death:	enemy, hole, fire, time-out	"BEWARE! THE FOLLOWING ARE DEADLY"	20
Avoid-danger	enemy, hole, fire	mentioned in deadly category	20

Figure 3. Goals and FLOs of the anlayzed segment of Super Mario Bros. 3 World 1 Level 1

movements necessary to accomplish them, defining the KLOs. KLOs are at a lower grain-size, and can be combined in different ways to accomplish many different FLOs. As in Card, et al., determining the sequence of operators at either of these levels constitutes a single GOMS analysis. Thus, determining both the sequence of FLOs and the sequence of KLOs is considered to be two GOMS analyses. However, since KLOs are simply split FLOs, the analysis process is to determine the appropriate FLO and then determine the KLOs that accomplish it; the analyses thus proceed in parallel.

The instruction booklet provides *methods*,[5] or sequences of KLOs, for accomplishing some FLOs (examples of methods from the instruction booklet are shown in Figure 4). Figure 5 lists the functional operators necessary to produce the observed behavior, the name of the methods provided by the instruction manual for accomplishing those FLOs, and the moves that make up the methods, and the KLOs that produce those moves. The exact KLOs making up a method depend on the relative position of Mario and the objects on the screen, so the keystrokes given are illustrative (e.g., a block may be to the right or left of Mario, so the KLO producing appropriate movement would be press-right-button or press-left-button, respectively).

[5] The instruction booklet uses the term *techniques* to describe what GOMS calls *methods*.

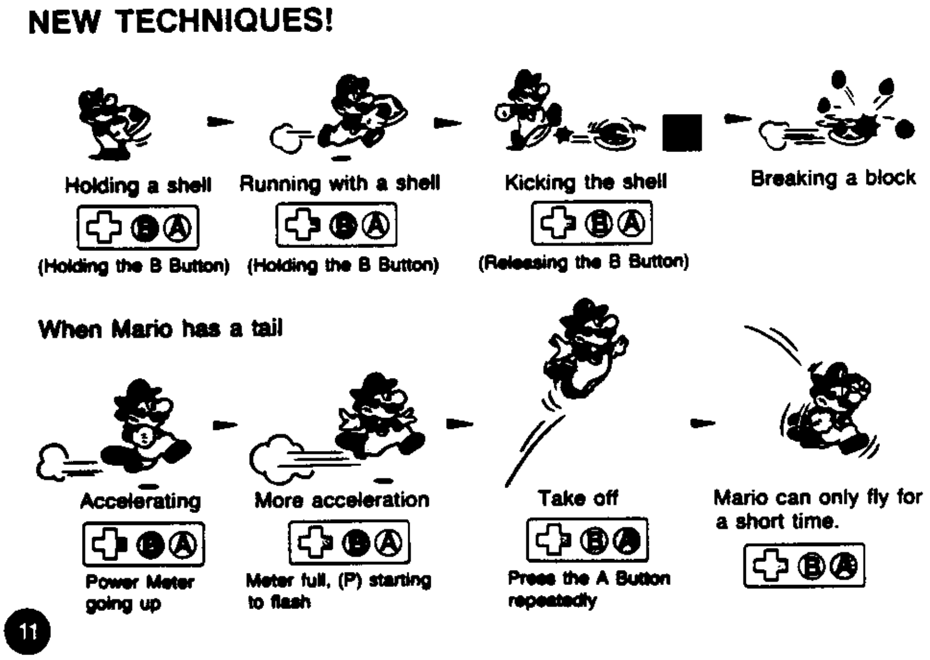


Figure 4. Methods suggested by the instruction booklet (from p. 11, *Super Mario Bros. 3. instruction booklet*, reprinted by permission of Nintendo of America, Inc.)

More than one method exists to accomplish each FLO used in the segment of the game studied (except for gather-item). We determined selection rules for these methods by analyzing two sources of information: the mechanics of the game itself (the necessary conditions for the methods to be applied), and the expert behavior observed in the videotape.

For the search-in-block FLO, the necessary conditions for the hit-from-bottom method are that the block be above Mario and that there be a clear path for him to jump up and hit it. For the hit-with-shell method, the necessary conditions are that there be a Koopa Trooper available to supply a shell and that there be a vantage point from which the shell can be thrown to hit the block. For the hit-with-tail method, Mario must possess a tail and the block must be vulnerable to tail-attack. In this segment of the game, none of the blocks are vulnerable to tail-attack, so this method is never observed. Because the hit-from-bottom-method has fewer steps it takes less time than the hit-with-shell methods; because it does not involve an enemy, it has less risk associated with it. Therefore, the selection rule we inferred is that, if the necessary conditions for the hit-from-bottom method exist, then the hit-from-bottom method will be selected. If not, then the hit-with-shell method will be selected. In this segment of behavior it is the case that if the necessary conditions for the hit-from-bottom method do not exist, the necessary conditions for the hit-with-shell method do exist, so this selection rule works. Reference to the expert's behavior did not contradict this inferred selection rule.

FLOs USED IN OBSERVED SEGMENT	POSSIBLE METHODS	CONDITIONS OF SELECTION RULES	MOVES IN METHOD	ILLUSTRATIVE KLOs TO ACCOMPLISH MOVES
search-in-block	hit from bottom	block is above Mario	move until under block	press-right-button
		path below block is clear	jump	press-A-button
	hit with shell	Koopa Trooper available	move to Kooppa	press-right-button
		vantage point available	immobilize Koopa	press-A-button
			pick up shell	hold-B-button
			move to block	press-right-button
			throw shell at block	release-B-button
	hit with tail	Mario has a tail	move to block	press-right-button
		block breakable via tail	hit block with tail	press-B-button
gather-item	move into		move to item	press-right-button
attack-enemy	stomp on	always applicable	move to enemy	press-right-button
		always selected	jump on enemy	press-A-button
	tail attack	Mario has a tail	move to enemy	press-right-button
		(not used by this expert)	hit with tail	press-B-button
avoid-danger	get out of way	a clear escape route exists	move away	press-right-button
		does not impede progress toward end of level		
	jump over	clear area above Mario's head	jump over	press-right-button
				press-A-button
		clear spot to jump to		
		"get out of way" impedes progress toward end		
	fly over	Mario has a tail	walk back	press-left-button
		clear runway	turn around	press-right-button
		clear take-off airspace	accelerate	press-right-button
		only when there		hold-B-button
		are no other options	take-off and fly	release-B-button
				press-A-button (repeatedly)
move-toward-end	walk	always selected	walk to the right	press-right-button
	run	not used	run to the right	press-right-button
				hold-B-button

Figure 5 - FLOs, methods, selection rules, and illustrative KLOs for the analyzed segment of the experts behavior.

For the attack-enemy FLO, the necessary condition for the tail-attack method is that Mario have a tail. When Mario does not have a tail, then there is no selection rule because only the stomp-on method is applicable, and this is the situation for the first enemy attacked. When Mario has a tail, the selection rule may be determined by analyzing the rewards and penalties associated with each method. For instance, the stomp-on method seems to require more precise timing of action and has more risk of dying for a novice than the tail attack. It is assumed that an expert can execute whichever maneuver he or she intends, so the risk component probably does not enter into the determination of the expert's selection rule. In the case of killing a ParaGoomba, a tail attack gets only 100 points whereas the two successive jumps required to kill the enemy using the stomp-on method get 100 and 200 points respectively, giving 200 more points than the tail-attack method. This must be traded-off against the extra time it takes to kill a ParaGoomba with two jumps, about 800 msec. For World-1 Level-1, the average rate of point collection is 150 points per second, so the extra 800 msec to kill the ParaGoomba by jumping is well worth the 200 extra points that method produces. Therefore, the selection rule for attack-enemy, when the enemy is a ParaGoomba is to use the stomp-on method exclusively. In the case of killing a Goomba, the selection rule is not so clear; the tail-attack and stomp-on method seem to have the same risk (small) and the same reward (100 points). However, the tail-attack method is described in the instruction manual as being a new technique for Super Mario Bros. 3. Since our expert was an expert in Super Mario Bros. 1 and 2 before this game, it is reasonable to assume a bias towards familiar methods, again giving a selection rule that uses stomp-on exclusively.[6] These selection rules were not contradicted by the expert's behavior in this segment.

For the avoid-danger FLO, the necessary condition for the get-out-of-the-way method is that there be a direction of escape, right, left, up or down. The necessary condition for the jump-over method is that there be clear space over Mario's head so he can jump. The necessary conditions for the fly-over method are that Mario has a tail, there is a clear runway to get up to speed and nothing overhead to block the take-off. The fly-over method takes so many more KLOs (to back up, accelerate and flap), that it should only be selected when there are no other options; this situation does not occur in the observed segment. The jump-over method takes one more KLO than the get-out-of-way method. However, if the get-out-of-way method is always taken when a danger is present, it may prevent forward progress in the game. For example, a pipe with a plant in it throwing fireballs at Mario may be to the right of Mario. The get-out-of-way method may send Mario running to the left until he is out of range of the fireballs. However, Mario would never progress beyond the fire-throwing plant with this selection rule in force, violating the clear-level goal and making Mario run out of time, thereby violating its own avoid-death goal. Therefore, the selection rule we infer is if the get-out-of-way method does not impede forward progress toward the end of the level, then use it, otherwise use the jump-over method for avoiding danger. The expert's behavior does not contradict this selection rule.

For the move-towards-end FLO, the methods are walk and run. The videotape did not to show clearly when the B-button was held down and when it was not. Also, the movement of Mario on the screen does not necessarily distinguish between these two methods without deeper knowledge of the mechanics of Mario's world (i.e. momentum). Therefore, we will not distinguish between these methods in our analyses, and just assume that move-towards-end is implemented by a press-right-button KLO.

[6] In the case of immobilizing a Koopa Trooper, the tail-attack method knocks the Koopa of the screen, so its shell is not available for use. Thus the attack of a Koopa Trooper embedded in the hit-with-shell method of the search-in-block FLO never uses the tail-attack method. This is one of the reasons why the attack of the Koopa Trooper embedded in the search-in-block FLO is not treated as a attack-enemy FLO.

The GOMS analysis process

Given these goals, FLOs, KLOs, methods and selection rules, analyzing the game involves perceiving the elements on the display and selecting the appropriate operators. First, a FLO is selected, then it is implemented by a series of KLOs, as described in Figure 6. The particular sequence of KLOs depend on the method used to implement the FLO and the relative position of Mario and the elements on the screen. The selection rules used to choose the appropriate method were described in the previous section and also appear in Figure 6. In the segment of behavior analyzed, these selection rules always result in a unique method, so there is no conflict resolution necessary for method selection. Therefore, at this stage in the analysis, we can uniquely determine the sequence of KLOs necessary to accomplish each FLO. This process of selecting FLOs constitutes the function-level GOMS analysis; that of selecting KLOs constitutes the keystroke-level GOMS analysis.

```
      search-in-block(x)
L1        is acceptable when block(x) is on the screen;
L2        has preference over any other operator (best);
L3        if there is no selected search-in-block(x) operator whose application is incomplete,
L4            and block(x) is closer to Mario than block(y),
L5            then search-in-block(x) is better than search-in-block(y).

      gather-item(x):
          is acceptable when item(x) is on the screen;
          has preference over any other operator (best);
          if there is no selected gather-item(x) operator whose application is incomplete,
              and item(x) is closer to Mario than item(y),
              then gather-item(x) is better than gather-item(y);
          if item(x) is moving and item(y) is not moving,
              then gather-item(x) is better than gather-item(y);
          if item(x) is moving and block(x) is on the screen,
              then gather-item(x) is better than search-in-block(x).

      attack-enemy(x):
          is acceptable when enemy(x) is on the screen and enemy(x) is killable;
          if enemy(x) is an immediate threat,
              then attack-enemy(x) is better than search-in-block, gather-item, avoid-danger, or move-towards-end;
          if enemy(x) is not an immediate threat,
              then attack-enemy(x) is worse than move-towards-end.

      avoid-danger(x):
          is acceptable when danger(x) is on the screen and danger(x) is invincible;
          if danger(x) is an immediate threat,
              then avoid-danger(x) is better than search-in-block, gather-item, attack-enemy, or move-towards-end;
          if danger(x) is not an immediate threat,
              then avoid-danger(x) is worse than move-towards-end.

      move-towards-end:
          is always acceptable;
          every other operator has preference over move-towards-end (worst).
```

Figure 6. Preferences for the proposal and selection of FLOs.

These analyses use a hand-simulation of what would happen if the GOMS models could interface directly with the game. That is, the analyst looks at the display, notes the elements visible, and uses the information in the preceding section to select an FLO and appropriate KLOs. Then the analyst presses the button corresponding to the next KLO to be performed. The button is only pressed until it has accomplished its function or until a display change occurs that creates the opportunity for a new FLO to be selected. For example, the hit-from-bottom method for the search-in-block FLO requires the analyst to move Mario horizontally until he is under the block to be searched. If the block is to the right of Mario, the analyst presses the A button until either Mario is under the block or another element appears on the display. This would be counted as one press-right-button KLO in the analysis. If no additional elements appear on the display before Mario reaches the block, the analyst releases the right-button (stopping Mario) then presses the A-button momentarily to make him jump straight up to break the block and land back below the block. The new display (now without the block just searched) is then analyzed to determine the next FLO and KLO, and the analysis cycle repeats.

In a Soar/GOMS model each FLO is *proposed* when certain elements are visible on the display. Each FLO has a set of *preferences* associated with it; proposing an operator is equivalent to making the preference for that operator *acceptable*. Other preferences may serve to resolve conflicts between competing FLOs. The conditions for proposal and the associated preferences are an integral part of the GOMS analyses, and, in this analysis, have been determined through a series of common-sense judgments about the task, described below.

Rules for proposing FLOs and preferences between competing operators are shown in Figure 6; a detailed explanation of the meaning of the preferences for the search-in-block FLO follows to help interpret that figure.

The search-in-block FLO is proposed (has an *acceptable* preference) any time there is an unexplored block on the display. In general, a search-in-block FLO is the best thing to do because it will often result in finding valuable items which give super powers, points, or money; this is stated in the instruction booklet, "Hit blocks...A useful item might pop out!". Thus, search-in-block is always given a *best* preference.[7] More than one unexplored block may be on the display at one time, so several search-in-block FLOs may be proposed, each evoked by one of the unexplored blocks, and each with a best preference. To resolve the conflict that arises from the existence of several best operators, other preferences are added to the operators, again based on task analysis, as follows.

The overall goal of attaining the most points possible dictates that the player be efficient in his or her actions, not only because running out of time kills Mario, but because any remaining time is converted into points at the end of the level. Thus, the block closest to Mario should be searched first. This is produced by making the search-in-block FLO for the closest block *better* than any of the other proposed search-in-block FLOs (L4 & L5 in Figure 6).

Occasionally, in the course of implementing a search-in-block FLO, Mario must move closer to an unexplored block (call it B) other than the current target (A). As they stand in the preceding paragraph, the preferences for search-in-block FLOs would change so that

[7] A *best* preference means that this operator is best if there are no other preferences that supersede best. However, other preferences can be in place that would make another operator *better* than the operator that is best, or *reject* the best operator, or create a tie with other best operators. For more details on the semantics of Soar preferences, see Soar 5 Manual.

search-in-block(B) would be better than search-in-block(A) as soon as Mario moves closer to block-B. This might be a reasonable strategy in many cases, exploring the closest block and going back to missed blocks later. However, it is easy to imagine arrangements of blocks that would cause the player to skip blocks and never get back to them because the presence of other blocks would continue to direct the preferences away from the skipped blocks. Thus, to get as many points as possible, the preferences must be adapted to prevent such target-switching when searching blocks. So the closer block can be better only when there is no existing search-in-block FLO that has not yet been accomplished (L3 in Figure 6). Reasoning similar to that presented here for the search-in-block FLO is used to construct the complete list of preferences shown in Figure 6.

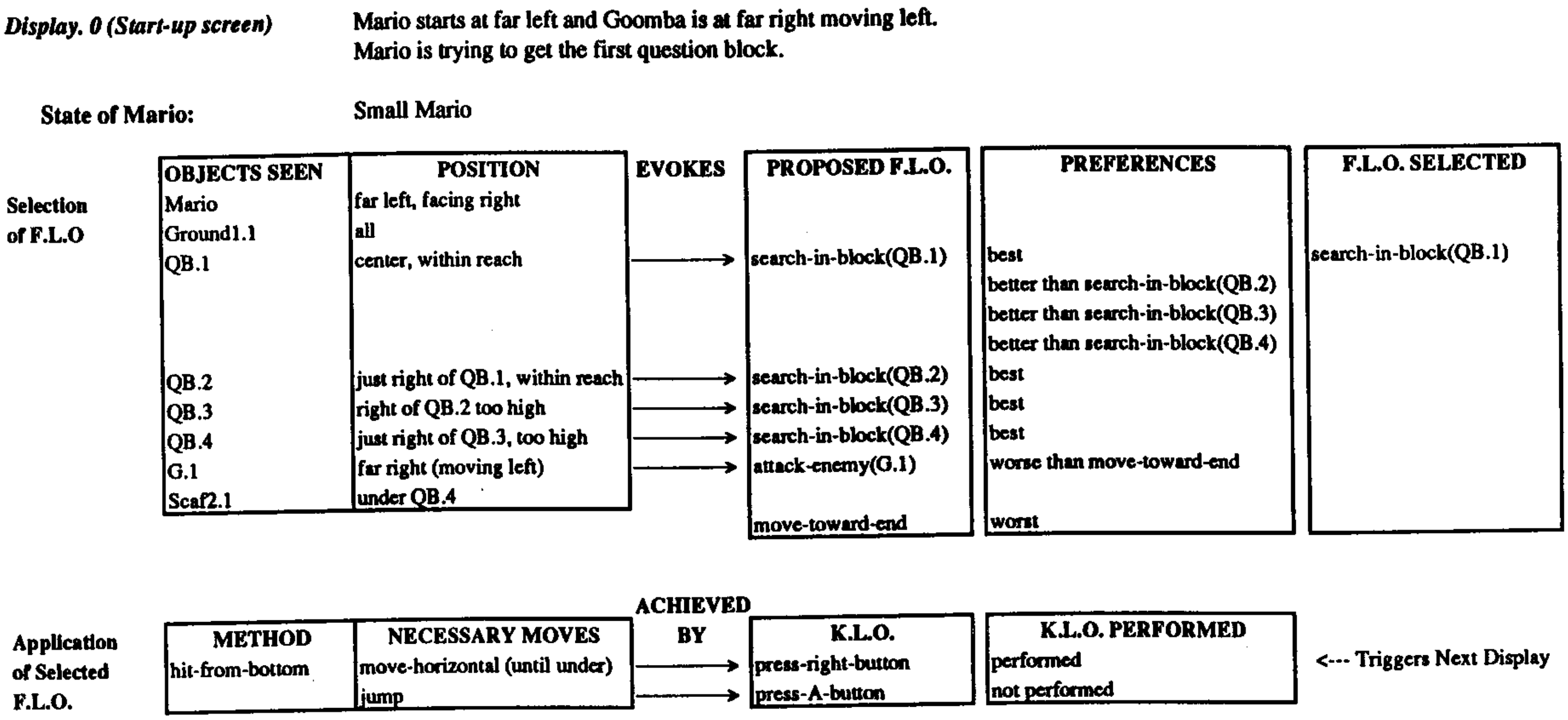
To illustrate the course of an analysis, refer to Figure 2, the first display of the game, and Figure 7, a chart describing the performance of the model. (Complete charts for the analyzed segment appear in Appendix II.) In the display, Mario is on the extreme left, on the ground. There are four unexplored question blocks visible (QB.1 through QB.4) and an enemy (called a Goomba and labeled G.1). QB.1 and QB.2 are within reach of a jump, whereas QB.3 and QB.4 are too high and Mario must jump onto the scaffold below QB.4 to be able to reach them. This information is appears in Figure 7, in the boxes labelled OBJECTS SEEN and POSITION.

Each QB and the Goomba cause an FLO to be proposed, according to the preferences in Figure 6, and recorded in Figure 7's box PROPOSED FLO. The position of the QBs and Goomba cause preferences to be installed, shown in Figure 7's PREFERENCES box. In this case, search-in-block(QB.1) has a best preference and is better than search-in-block(QB.2), search-in-block(QB.3), and search-in-block(QB.4) because it is closest to Mario. Since the Goomba is on the far right of the screen, it is not an immediate threat to Mario and the preferences make attack-enemy(G.1) worse than the default FLO, move-towards-end (always acceptable, but worst). Given the semantics of Soar preferences, search-for-block(QB.1) is selected as the FLO to implement; this appears in the FLO SELECTED box.

The selected FLO is then implemented with a method comprised of several KLOs. The appropriate method is determined with the selection rules of Figure 5 and appears in the METHOD box in Figure 7. The moves that Mario has to make to accomplish that method are listed in the NECESSARY MOVES box. These moves are achieved by specific KLOs listed in the KLO box. As previously described, these KLOs are performed by hand by the analyst to determine their effects on the game's display.

Since this game is highly interactive, the situation may change before all the KLOs necessary to implement a specific FLO can be performed. This happens at the very beginning of the game; as the player presses the right-button to move Mario under QB.1, the Goomba runs to the left and becomes an immediate threat before Mario reaches QB.1. When the Goomba becomes an immediate threat, the preferences for FLOs change and must be reassessed. Thus, the press-right-button KLO is performed, but the press-A-button KLO (which would make Mario jump into the QB) is not performed before the situation changes sufficiently to interrupt the implementation of search-in-block(QB.1). Figure 7 records this information in the KLO PERFORMED box and indicates which KLO triggers the next display that causes a change in preferences to occur.

Appendix II contains charts like the one in Figure 7 that follow from the FLOs, methods, selection rules, preferences, and KLOs described above. To create those charts, we started with the initial display of the game, and moved Mario one KLO at a time until the elements on the screen changed sufficiently to require reassessment of preferences. That is, at the initial display, we pressed the right-button until either Mario was under QB.1 (where the



Display. 0 (Start-up screen) Mario starts at far left and Goomba is at far right moving left.
Mario is trying to get the first question block.

State of Mario: Small Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	far left, facing right				
Ground1.1	all				
QB.1	center, within reach	→	search-in-block(QB.1)	best better than search-in-block(QB.2) better than search-in-block(QB.3) better than search-in-block(QB.4)	search-in-block(QB.1)
QB.2	just right of QB.1, within reach	→	search-in-block(QB.2)	best	
QB.3	right of QB.2 too high	→	search-in-block(QB.3)	best	
QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	
G.1	far right (moving left)	→	attack-enemy(G.1)	worse than move-toward-end	
Scaf2.1	under QB.4				
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-from-bottom	move-horizontal (until under)	→	press-right-button	performed	<--- Triggers Next Display
	jump	→	press-A-button	not performed	

Figure 7. Example of the charts in Appendix II explaining the predictions of the models

method for implementing search-in-block(QB.1) would require a press-A-button KLO) or something else happened to change the preferences. In this case, the Goomba became an immediate threat before Mario was under QB.1, so another chart was created (Display.1 in Appendix II).

To see the sequence of FLOs predicted by the model, read the entry in each chart's FLO SELECTED box. To see the sequence of KLOs predicted by the model, read the entries in each chart's KLO box that are marked *performed* in the KLO PERFORMED box.

RESULTS OF GOMS ANALYSIS

We compared the model's predictions to the observed performance of the expert, KP, using the charts in Appendix II and the transcription of KP's performance in Appendix I. KP's verbal protocol lagged his performance and in some cases later in the game he had to stop speaking altogether to continue to play the game. We believe this indicates that the verbal report of his goals was treated as a secondary task, dropping out when the primary task of playing the game was too difficult, and may not be complete. Therefore, our comparison uses only what KP makes Mario *do* rather than what KP *says* he is making Mario do.

Appendix III contains charts that record the comparison; an example for the first screen display appears in Figure 8. At the top of the chart, Display.x corresponds to the Display.x in the model charts (Appendix II).[8]

In the comparison charts, the top row of boxes, OBJECTS SEEN and POSITION, contains the information about what is on the screen at this point in the videotaped game and correspond closely to the same boxes in model charts. The correspondence between expert and model is not exact, because the model and the expert may not be in exactly the same position at every display change. The second row of boxes list the OBSERVED BEHAVIOR, the INFERED FUNCTION of that behavior, and INFERED FINGER ACTION. The inferred function is *ambiguous* if more than one function could be served by that behavior (e.g. moving to the right could move Mario towards the end of the level, or it could bring him under a block so he can search it), and listed as a particular function if only one function is obviously served (e.g. jumping into a QB unambiguously serves to search that QB). The finger action was inferred from the behavior because it was not directly observable from the videotape, but most movements of Mario can only be produced by specific button presses, so the inference is uncontroversial. The third row of boxes lists the FLOs that would be consistent with the observed behavior, the specific FLO predicted by the model, and the KLO that is predicted by the model. The FLO predicted by the model can be compared directly to the inferred function of the observed behavior in the box above it. The KLOs can be compared directly to the inferred finger actions in the box above it.

Figures 9 and 10 graphically present the comparisons between the observed expert behavior and the model's predictions. Figure 9 shows the comparison at the function-level and Figure 10 shows the comparison at the keystroke-level. In both figures, the operator names are listed along the vertical axis. The horizontal axis is labelled with the display number

[8] There are more charts for the model predictions than for the expert comparisons because the expert moves in a fluid, continuous motion while the model simulates Mario walking one step at a time, stopping before and after jumps. Some of the display changes occur only when Mario is moved beyond a specific place in the world. The model's incremental movement produces some displays where the next elements has not yet appeared and the only applicable FLO is move-toward-end. In contrast, the expert's fluid movement moves Mario beyond the triggering point in service of the previous function, so the displays where nothing new presents itself never occur.

Display. 0 (Start-up screen)

Mario starts at far left.
Goomba is at far right moving left.
Mario is trying to get the first question block.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	far left, facing right
Ground1.1	all
QB.1	center, within reach
QB.2	just right of QB.1, within reach
QB.3	right of QB.2 too high
QB.4	just right of QB.3, too high
G.1	far right (moving left)
Scaf2.1	under QB.4

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario moves right	ambiguous	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.1)	search-in-block(QB.1)	press-right-button
search-in-block(QB.2)		
search-in-block(QB.3)		
search-in-block(QB.4)		
attack-enemy(G.1)		
move-to-end		

Figure 8. Example of the charts in Appendix III comparing the expert's observed behavior with the predictions of the models.

corresponding to the Display.x label on each of the model charts in Appendix II. This designation corresponds roughly to time, where each display represents about a second's worth of activity.

In Figure 9, for each display, a black dot marks the inferred function of the observed behavior (from the INFERED FUNCTION box of Figure 8). If no unique function can be inferred from the observed behavior, no black dot appears in that display column. A gray dot marks the FLO predicted by the model (from Figures 7 & 8). Thin black circles indicate the FLOs consistent with the observed behavior (from Figure 8). To highlight the predicted sequence of FLOs, the predicted FLOs (gray dots) are joined by gray lines; to highlight the

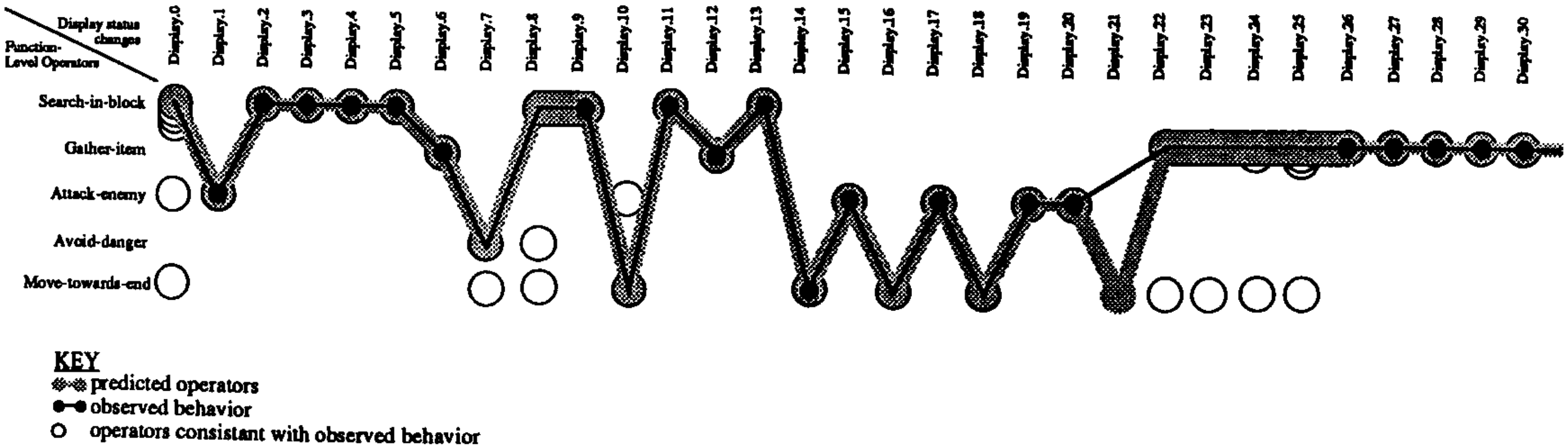


Figure 9. Comparison of the function-level model predictions with the observed behavior.

Black dots indicate functions inferred from the observed behavior. For instance, when Mario hits a block, we inferred that the intended function was to search in that block. Gray dots indicted the predicted FLOs. When an FLO is predicted but a corresonding function cannot be unambiguously inferred from the observed behavior, a gray dot appears, but no black dot is drawn in the same Display column. In that case, all the FLOs consistant with the observed behavior are indicated with thin black circles. Gray lines connect the predicted FLOs and black lines connect the inferred functions. When no function can be unambiguously inferred, the black line passes through the predicted FLO if that FLO is consistant with the behavior, and does not pass through the predicted FLO if that FLO is contradictory to the observed behavior. Elongated gray circles represent consecutive displays in which the same FLO, with the same argument, is selected. Elongated black circle indicate that the behavior observed throughout those displays is inferred to be in the service of the same function.

the redundant FLOs making it look like one elongated circle of an FLO (Displays.8 & .9, and Displays.23 to .27). When a function cannot be inferred from the observed behavior and no black dot appears in the display column, the black line is passed through the gray dot if the observed behavior is consistent with the predicted FLO; if they are inconsistent, the black line does not pass through the gray dot in that display column, but goes straight to the next inferred function.

The sense of this graphic display is that when the black and gray lines track each other, the model is making good predictions of observed behavior. Where the black and gray lines do not track, the model is not predicting the observed behavior. Where there is a haze of thin black circles above and below the black dots, the behavior is consistent with many possible functions and it is likely that any model prediction will track behavior; where black circles are few, the observed behavior is consistent with only a few functions and the model must be accurate to track behavior.

Figure 10 uses the same graphic conventions as Figure 9, but for the keystroke-level behavior and predictions. Each horizontal unit, delineated by dotted lines, corresponds to the FLO implemented by the KLOs shown in that unit and also to the display that caused the selection of that FLO. Therefore, each horizontal unit is labelled at the top with its corresponding Display.x and at the bottom with its associated FLO.

DISCUSSION OF THE GOMS ANALYSIS

Some caveats

Before discussing the implications of this analysis, three strong cautions must be presented. First, this analysis explores a short segment of behavior. Although a qualitative look at an additional segment twice as long as the current analysis, representing traversal of the rest of World 1 Level 1, indicates that much of that behavior is also predictable, the detailed analyses and comparisons must be done before any conclusions can be drawn with confidence.

Second, World 1 Level 1 is the simplest level in the game. Although it is not evident in the expert videotape, if Mario stops moving, nothing will happen most of the time at this level of play. Thus, the interaction is primarily user-driven, not system-driven. This feature is not common in many other types of video games and it does not even persist in Super Mario Bros. 3, as the higher worlds have many more enemies actively seeking to kill Mario.

Last, in this part of World 1 Level 1 the objective of the next cycle is almost always visible in the display. Although a FLO that searches through the world for blocks to break or items to gather can be deduced from the instruction booklet, it was not invoked in this segment of behavior because the next block to break, item to gather, or enemy to attack, was visible in 26 of the 31 displays. Other games, and other levels even within this game, are not as perceptually driven. Without constant triggering of operators by the elements in the display, the behavior would take on a more searching aspect (in the case of no knowledge), or a more planful aspect (in the case of additional knowledge of hidden items).

The nature of expertise in these analyses

Before examining the specifics results of these analyses, it is useful to locate the use of expertise in both the models and the expert's behavior. The GOMS models have no knowledge of Mario's world; they do not know anything except what is on the screen.

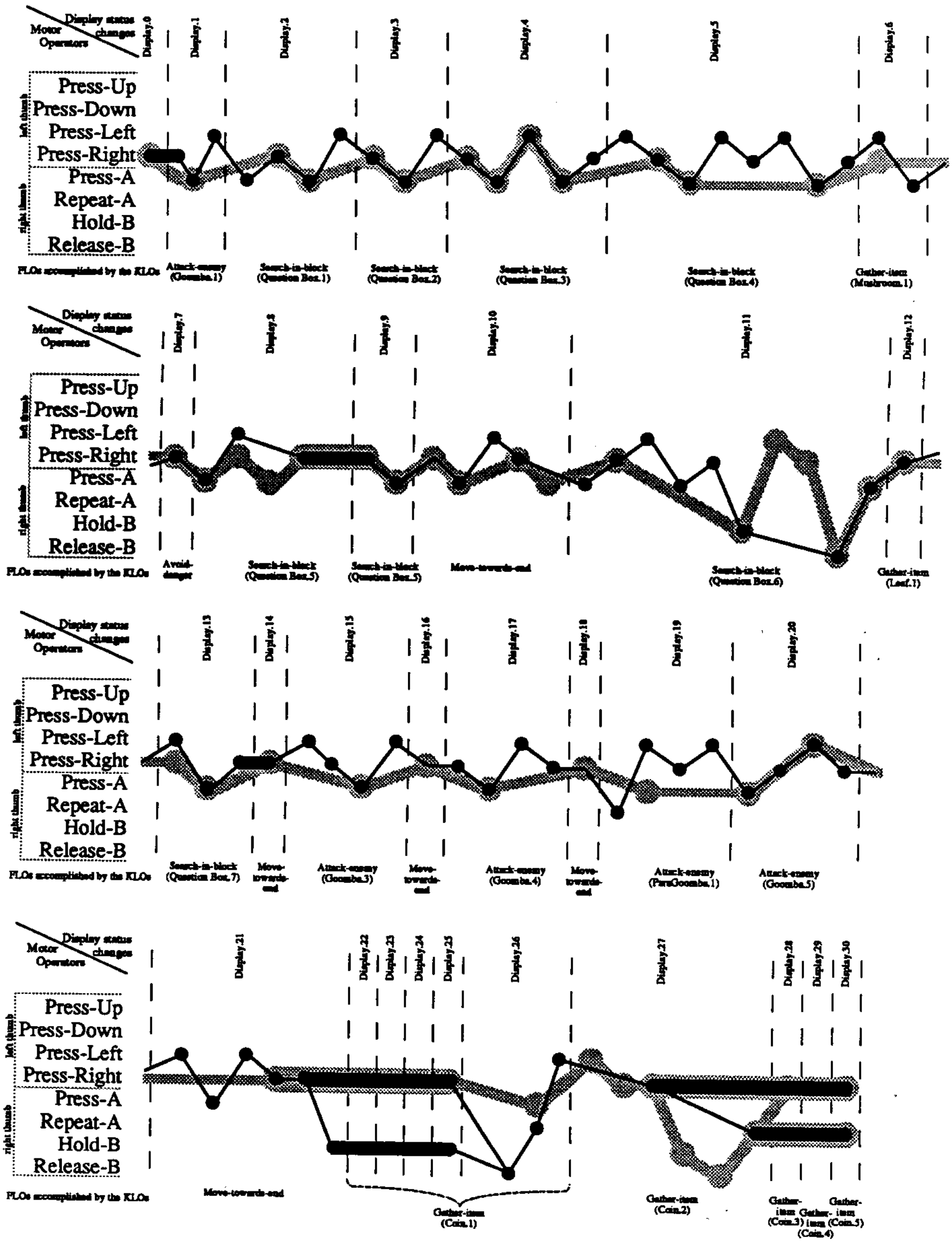


Figure 10. Comparison of keystroke-level predictions to observed behavior
(description same as that of Figure 9.)

Therefore, they cannot anticipate either functions or keystrokes. This is not true of the expert. A post-trial interview with KP, as well as observation of several trials of play, clearly showed that he knows every detail of World 1 Level 1. He knows where the treasures are hidden and what super powers they bestow. He knows when and where enemies will appear and whether each enemy should be killed or whether it can be ignored. His actions in two situations indicate that he anticipates functions to perform with elements not yet on the display. These situations are when he jumps to catch the mushroom before it emerges from QB.4 and when he starts his take-off to fly along a path of coins leading into the sky, before that path appears on the display. It is not surprising that an expert has such knowledge, only that this world requires him to use it so infrequently. The models make predictions without reference to it at all.

One the other hand, the GOMS models are expert in their execution of operators. KLOs are always assumed to accomplish their intended movement. For example, if a press-A-button KLO is used to jump on the back of an approaching enemy, it succeeds, never missing and allowing the enemy to kill Mario. This is clearly not the case with a human novice who has never used the Nintendo hand-held controller before. Informal observation of novice players shows many errors in execution: jumps are not timed correctly or are not initiated in the correct place so Mario doesn't kill an enemy or misses a block, buttons are not released in time so Mario runs off the cliff, etc. In addition, the selection rules in these models embody expert knowledge: the models always select the same method for accomplishing an FLO the expert selects. Since KLOs are always successful, and the method always reflect an expert selection, a sequence of KLOs in service of an FLO is always successful if it completed before another FLO is selected.

Thus, these GOMS models represent the behavior of a player with expert motor movements, expert knowledge of methods for collecting treasures and killing enemies, but no knowledge of what they will encounter in this world. This description coincidentally fits a human player who has played Super Mario Bros. and Super Mario Bros. 2., but never Super Mario Bros. 3. Such a person would have be expert in most methods and all keystrokes, but would not know the world at all. Further test of these models would benefit from observation and analysis of such a player.

Highlights of these analyses

The results of this analysis indicate that GOMS can capture the knowledge necessary to predict the course of this behavior. The FLOs in this segment of behavior are virtually dictated by some simple heuristics derived from the overriding goals of the game, the operators and methods described in the instruction booklet, and the elements visible on the display. Of the 31 FLOs predicted, 21 can be unambiguous inferred from the observed behavior, 9 are consistent with observed behavior, only 1 is inconsistent with observed behavior, and no behavior indicates any functions not predicted by the model. At the keystroke-level, of the 62 KLOs predicted, 46 are observed, 3 are consistent with observed behavior, 12 are inconsistent with observed behavior, and 35 keystrokes are observed but not predicted.

Figure 9, the comparison of the observed behavior to the predicted FLOs show some interesting features of the analysis. Most striking is the excellent agreement between the predicted FLOs and the observed behavior. Only once in 31 opportunities is an FLO selected that is inconsistent with observed behavior (Display.21). This instance is at a point in the game where the expert seems to anticipate gathering some coins that are not yet visible on the screen and for which he must attain flying speed. He moves left to get a running start for take-off. The model does not have the prior knowledge that coins will eventually appear to the right, and moves towards the end of the level (to the right) until it sees the coins, and then gathers them. Thus, the expert moves to the left in anticipation of flying, while the model moves to the right until it sees something to gather. As soon as the model sees the coins, the model and the expert have the same knowledge and they come back into synch.

A second interesting feature is that in 4 of the 11 displays in which the expert behavior does not match the predicted behavior (Displays.10, 16, 18, and 21), the predicted FLO is move-towards-end. This is the default FLO used when there are no elements on the display to trigger the proposal of other FLOs. In effect, the model is exploring the world when it selects this FLO. The expert, unlike the model, knows this part of the world quite well and anticipates the upcoming elements. The expert does not display this exploring behavior, but anticipates the next function and starts to perform it immediately. Thus, if the expert's cognitive processes were more clear from the verbal protocol, we expect more goal-directed behavior to emerge than is predicted by the model. However, if a novice were to play the game, with no knowledge of what comes next (like the model), the model predicts that this exploring behavior would be evident.

A third interesting feature is that only 8 of the 31 displays had observed behaviors consistent with several FLOs simultaneously (Displays 0, 7, 8, 10, 23, 24, 25, and 26). This means that the predicted FLOs must be exactly in line with the expert's goals to make a correct prediction. The behavior can be interpreted only as belonging to a single function most of the time, so picking an FLO with incorrect preferences, or at random, would rarely result in a match to observed behavior. This indicates that the function-level model is quite good at predicting the sequence of functions an expert will perform in this phase of the game.

The comparison of keystrokes inferred from the observed behavior and the KLOs predicted by the keystroke-level model, Figure 10, also has several interesting features. The most striking is the dramatic difference in predictive power between the function-level and KLO levels; the function-level predictions are almost perfect while the keystroke-level model predicts only about half of the observed behavior (46 of the 96 observed keystrokes). This is the same pattern of results obtained in Card, Moran, and Newell's analyses of text editing (1983), with FLOs predicting close to 100% of text-editing behavior and KLOs predicting about 60%.

Such a large drop in predictive power indicates that the keystroke-level models, both for the text-editing task and the game-playing task, may be missing important features that correspond to the unexplained behavior. For the text-editing task, Card, Moran, and Newell state that much of the unexplained behavior involved hand movements outside the model, e.g. the user licked her fingers before turning each page, an act not included in their model. In this analysis, there seems to be a similar type of unexplained behavior, twisting while jumping.

Twenty-four of the 37 observed keystrokes not predicted involve turning Mario left or right while jumping or floating. This twisting motion may be in service of a goal not represented in this model, e.g. displaying flashy behavior to impress other players. This hypothesis directs research attention to the social aspects of video game play to discover goals not evident from the instruction booklet. Alternatively this behavior may serves to slow Mario's horizontal motion while in the air. The second hypothesis, which we have recently been assured by other experts is the correct explanation, indicates that the motor model used in this analysis is too simple to capture the interaction of forces (e.g. the equivalents of gravity, friction, and air resistance) in Mario's world. A more detailed perceptual-motor model should include duration of button presses, perceptual monitoring, and the player's prediction of effects.

Questions of learning

Production-system formulations of GOMS have been used successfully to predict learning of several domains (e.g., Kieras & Bovair, 1986; Ziegler, et al, 1986; Lee, et al., 1989; Singley & Anderson, 1989), and may be useful for understanding learning in this domain as well. The learning predicted in the previous studies involved reading instructions and internalizing performance rules based on those instructions, plus practice with error feedback. That process describes the learning necessary to perform the actions and methods described in the instruction booklet. However, several other types of learning are evident in this interaction.

There is a strong component of motor learning; a novice continuously overshoots or undershoots runs and jumps, while an expert manipulates Mario more precisely. GOMS is not currently capable of predicting motor learning.

The other obvious learning behavior is "learning-by-dying", that is, Mario dies because of strategic mistakes and the player learns to avoid those mistakes in the future. For instance, the method of kicking a Koopa shell into a block on the ground to break it open is described in the instruction manual. When a novice performs this action on QB.6, the Koopa shell bounces off the block, right into Mario. This event happens too quickly for the immediate interaction cycle to produce adaptive behavior, and Mario dies. The novice then learns to modify the method to include jumping out of the way as soon as Mario kicks the shell. If the conditions surrounding each death are added to the condition of a production rule, and are combined with the appropriate avoidance tactic as the action of that production rule, new rules could be added to the knowledge in the GOMS analysis. Previous work with Soar, modeling strategy change in a developmental task (Newell, 1990) and modeling recovery from errors (Laird, 1988) indicates that the architectural mechanisms are sufficient to learn such knowledge when the learner can set the pace of learning. This situation raises the question as to whether those same mechanisms will be able to learn when the feedback occurs within the immediate interaction cycle. Another question is whether a regularity similar to the 30 seconds per operation[9] found in previous studies of learning and transfer would emerge from this interactive style of learning.

Other forms of learning produce behavior in the service of elements not yet visible on the display (discussed in the comparison between observed functions and predicted FLOs). At least two types of learning might occur to produce this behavior. The expert seems to anticipate elements before they are visible, so he might have learned cues that are visible prior to the elements of interest. Alternatively, KP says "get the ... mushrooms so you can turn into Super Mario", an indication of a goal above the FLO gather-item(mushroom); such higher level goals might serve to combine FLOs into methods themselves. These conjectures raise many questions. What knowledge structure is necessary to store and access goals in advance of directly relevant elements on the display? What role might a hierarchy of goals above the FLOs play in anticipatory behavior. How can the rapid interaction observed here, with a new FLO occurring every second and behavior being so time-critical and difficult that a verbal protocol cannot keep up, allow processing of the behavior to create these higher level goals, or store cues for anticipation of goals? Does such processing happen in the course of the game, or is there only learning-by-dying

[9] Soar *operators* are equivalent to the *productions* used in other research. Soar *productions* are at a lower grain-size and would not be expected to correspond to the learning times found previously.

(where the user reflects on what happened only when the immediate interaction cycle is broken by Mario's death), or perhaps learning occurs only even further outside the confines of the game in discussion with other players or while reading "Nintendo Power"[10] (a magazine containing game hints)? How can Soar's chunking mechanism produce this anticipatory behavior?

An interesting prediction about what is and is not learned arises from the environmentally driven characteristic of this portion of the game. Since expert knowledge about upcoming elements in the world is not used in the selection of FLOs,, different experts would follow the same FLO sequence, and even novices would produce the same function-level behavior. The motor operations would differ, especially for novices who are not able to manipulate Mario reliably, but the functions they try to accomplish (e.g. search-in-block(QB.1) then search-in-block(QB.2)) would not be a learned sequence, rather they would arise at all levels of skill from the elements on the display. Although verification of this prediction remains an open issue, informal observation of an additional expert and several novices do not contradict this prediction.

Speculation about motivation

Perhaps the most striking feature of video games is that people play them for hours on end. Empirical studies of computer games have produced speculation about the factors contributing to motivation (Malone, 1980), but as yet GOMS has not been used to explore this aspect of user-computer interaction. Based on the form and process of this GOMS analysis, we present some purely speculative words about why this game is so absorbing. First, the perceptually-based production system form of this model allows for the notion of *capture*. That is, since cues in the environment trigger operator proposal and selection, whenever those cues are present, those operators will be selected and executed. Thus, the environment captures the player and sends him or her looping through the immediate interaction cycle. Second, the rate of goal satisfaction (i.e., FLO accomplishment) may be related to this game's ability to absorb its players. If the rate is too high, the game is too easy and boring (e.g., KP yearned to go on to a higher level); if the rate is too low, the game becomes frustrating; if it is just right, the game captures players for hours on end. Thus, other open research questions are whether these concepts of capture and rate of goal satisfaction have any relation to what we know as fascination, or motivation, and, if so, whether the parameters of these concepts can be discovered and eventually manipulated to predict the fascination level of new games, educational programs, and other application interfaces.

Directions for future research

The stated purpose of the original panel, and this report, is to demonstrate GOMS analyses in the area of highly interactive tasks and encourage researchers and practitioners to use GOMS in their work. We see three directions in which future work may spring from our analyses.

The first direction is the extension of this particular analysis. The panel presentations represented a first pass GOMS analysis for each of the participants. This paper represents a "pass and a half" for our analyses, i.e., cleaned up but not substantially changed from the first pass. A second pass would tackle issues raised by the panel participants: perceptual monitoring of the effects of motor operations, a more detailed model of motor operators and their interactions with the forces designed into Mario's world, examination of the operator

[10] Nintendo Power is a trademark of, and is under copyright to, Nintendo of America.

durations and comparison to the human time course of behavior. A second pass would also include consideration of other research in the domain, notably the AI programs that play the video games Pengo and Amazon (Pengi in Agre & Chapman, 1987, and Sonja in Chapman, 1990, respectively). Although these programs were never intended to be cognitive models of human behavior, and were never formally compared to human behavior, they provide insight into a simple architecture that uses relatively little game knowledge but is sufficient for performing the immediate interaction necessary to play the game.

A second direction is for HF practitioners to use GOMS to make predictions of human behavior with the systems they help to design. At first, these analyses may be done in parallel with other HF methods, e.g. user trials, to validate the analytic predictions (see Gray, et. al. 1990 for an example). In time, we expect that confidence in GOMS analyses will grow and allow design decisions to be made in advance of the working prototypes (or Wizard-of-Oz mock-ups) and extensive user testing.

The third direction is to tackle the limitations of GOMS and extend the analysis technique itself. As presented above, GOMS is an evolving analytic technique. Its original formulation presented by Card, Moran and Newell (1980, 1983) has undergone several changes to extend the original scope, most recently this includes a change in the underlying architecture to the Soar unified theory of cognition. The change to Soar has allowed GOMS to encompass the interruption of tasks and responsiveness to the environment (as begun in this analysis and in John, et. al., 1990). We expect this change will lead to increased understanding and prediction of learning, errors, knowledge representations, and other issues of interest to the HF and HCI community, as well as to the cognitive science and cognitive psychology communities.

CONCLUSIONS

This analysis demonstrates GOMS's power to predict behavior in a highly interactive domain. It shows that GOMS is especially useful for making predictions of human behavior at the function level. It demonstrates how even an initial GOMs analysis can help direct research effort by focusing on areas of the model that do not predict human performance well. Further, it can make predictions about the general character of a task, e.g., that novices and experts alike will have the same function-level goal structure in this environmentally driven interaction.

This preliminary investigation also poses many questions. In pursuit of answers to those questions, this domain seems rich in behavioral phenomena, affording opportunities to explore the immediate interaction cycle with its rapid perception, cognition and motor actions, several aspects of learning, and perhaps even the elusive concept of fascination. In addition, research into the validation of GOMS analysis in real-world settings and the extension of the GOMS analytic technique would be valuable contributions to HF and HCI methodologies.

ACKNOWLEDGEMENTS

We would like to thank Kyle Gray, Jay Anderson, and Billy Esch for their expert advice about playing Super Mario Bros. 3, and Gin Peck and Carolanne Fisher for their comments on drafts of this paper.

REFERENCES

Agre, P. E. & Chapman, D. (1987) Pengi: An implementation of a theory of activity. *Proceedings of the AAAI'87: National Conference on Artificial Intelligence.* Menlo Park, Calif.: American Association for Artificial Intelligence.

Anderson, J. R. (1983). *The architecture of cognition.* Cambridge, Mass: Harvard University Press.

Card, S. K., Moran, T. P., & Newell, A. (1980) Computer text-editing: An inforamtion-processing analysis of a routine cognitive skill. *Cognitive Psychology, 12,* 32-74.

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

Chapman, D. (1990) *Vision, Instruction, and Action.* (Tech. Rep. No. AI-TR 1204). Cambridge: MIT, Artificial Intelligence Laboratory.

Gray, W. D., John, B. E., Stuart, R., Lawrence, D., Atwood. M. E. (1990) GOMS meets the phone company: Analytic modelling applied to real-world problems. *Proceedings of Intertact '90 Third IFP Conference on Human-Computer Interaction.* Cambridge, England.

John, B. E. (1988) *Contributions to engineering models of human-computer interaction.* Doctoral dissertation, Carnegie Mellon University.

John, B. E. (1990) Extensions of GOMS analyses to expert performance requiring perception of dynamic visual and auditory information. In proceedings of *CHI, 1990* (Seattle, Washington, April 30-May 4, 1990) ACM, New York, 107-115.

John, B. E. & Newell, A. (1987) Predicting the time to recall computer command abbreviations. In proceedings of *CHI+GI, 1987* (Toronto, April 5-9, 1987) ACM, New York, 33-40.

John, B. E. & Newell, A. (1989) Culmulating the science of HCI: From S-R compatibility to transcription typing. In proceedings of *CHI, 1989* (Austin, Texas, April 30-May 4, 1989) ACM, New York, 109-114.

John, B. E., Newell, A., Card, S. K. (1990, February) *Browser-Soar: A GOMS-like model of a highly interactive task.* Talk presented at the Human Computer Interaction Winter Workshop, San Diego, California.

John, B. E., Rosenbloom, P. S., & Newell, A. (1985). A theory of stimulus-response compatibility applied to human-computer interaction. In proceedings of *CHI, 1985* (San Francisco, California, April 14-18, 1985) ACM, New York, 212-219.

Kieras, D. E. (1988) Towards a practical GOMS model methodology for user interface design in M. Helander (ed.) *Handbook of Human-Computer Interaction.* Elsevier Science Publishers B. V. (North-Holland).

Kieras, D. E., & Bovair, S. (1986) The acquisition of procedures from text: A production-system analysis of transfer of traning. *Journal of Memory and Learning*, 25, 507-524.

Kieras, D. E. & Polson, P. G. (1985) An approach to the formal analysis of user complexity. *International Journal of Man-Machine Studies, 22*, 365-394.

Laird, J. E. (1988) Recovery from incorrect knowledge in Soar. *Proceedings of the AAAI'88 National Conference on Artificial Intelligence*. Menlo Park, Calif.: American Association for Artificial Intelligence.

Laird, J. E., Newell, A., & Rosenblom, P. S. (1987) Soar: An architecture for general intelligence. *Artificial Intelligence, 33*, 1-64.

Laird, J. E., Swedlow, K., Altmann, E., & Congdon, C. B. (1990) *Soar 5 User's Manual*. Carnegie Mellon University.

Lee, A. Y., Polson, P. G. & Bailey, W. A. (1989) Learning and transfer of measurement tasks. In proceedings of *CHI, 1989* (Austin, Texas, April 30-May 4, 1989) ACM, New York, 115-120.

Lerch, F. J., Mantei, M. M., and Olson, J. R. (1989) Skilled financial planning: The cost of translating ideas into action. In proceedings of *CHI, 1989* (Austin, Texas, April 30-May 4, 1989) ACM, New York, 121-126.

Lewis, R. L., Huffman, S. B., John, B. E., Laird, J. E., Lehman, J. F., Newell, A., Rosenblom, P. S., Simon, T., & Tessler, S. G. "Soar as a Unified Theory of Cognition: Spring 1990. in the *Proceedings of the Twelfth Annual Conference of the Cognitive Science Society*, July, 1990.

Malone, T. W. (1980, August) *What makes things fun to learn? A study of intrinsically motivating computer games*. (Res. Rep. SSL-80-11). Xerox, Palo Alto Reasearch Center.

Newell, A. (1990) *Unified theories of cognition*. Cambridge, Mass: Harvard University Press.

Newell, A. & Card, S. K. (1986) Straightening out softening up: Response to Carroll and Campbell. *Human Computer Interaction*, 2, 251-267.

Newell, A. & Card, S. K. (1985) The prospects for psychological science in human-computer interaction. *Human Computer Interaction, 1*, 209-242.

Nintendo of America, *Super Mario Bros. 3 Instruction Booklet.*

Olson, J. R. & Nilsen, E. (1988) Analysis of the cognition involved in spreadsheet software interaction. *Human Computer Interaction, 3*, 309-350.

Olson, J. R. & Olson, G. M. (1990) The growth of cognitive modeling in human computer interaction since GOMS. *Human Computer Interaction, 5*, 221-266.

Polson, P. G., & Kieras, D. E. (1985). A quantitative model of learning and performance of text editing knowledge. In proceedings of *CHI, 1985* (San Francisco, California, April 14-18, 1985) ACM, New York, 207-212.

Rosenbloom, P. S. (1983) *The Chunking of Goal Hierarchies: A Model of Practice And Stimulus-Responsé Compatibilty*. Ph.D. diss., Computer Science Department, Carnegie Mellon University, Pittsbugh, PA.

Rumelhart, D. E., McClelland, J. L., & the PDP Research Group. 1986. *Parallel Distributed Processing: Explorations in the microstructures of cognition. Vol 1: Foundations*. Cambridge, Mass. MIT Press.

Singley, M. K. & Anderson, J. R. (1989) *The transfer of cognitive skill.* Cambridge, Mass: Harvard Univeristy Press.

Ziegler, J. E., Hoppe, H. U., & Fahnrich, K. P. (1986) Learning ans transfer for text and graphics with a direct manipulation interface. In Mantei and P. Orbeton (eds.), *CHI'86 Human Factors in Computing Systems* (Boston, April 13-17). New York: ACM.

Appendix I - Transcription of Observed Behavior

Transcribed by Wayne Gray, 7/31/90
Tape of S#1-KP
Take 5 WITHOUT eye scan, with keypad.
Tape sequence begins at 10:15:73, note that this refers to the large numbers at the bottom right (you may need to adjust the overscan on your monitor to see these clearly).

Auditory stuff done with the help of the Speech Lab, Sara Basson, et al.

Transcript supplied by Wayne Gray				*Inferences made during our analysis*	
TIME (SEC.)				INFERRED	CORRESPONDING
START	END	VOICE	SUPER MARIO'S MOVEMENTS	FINGER ACTION	DISPLAY
16.23	17.23	OK, Anders ready			
19.12	20.02	I'm going to the first			
20.07	20.34	level			
20.42	21.54	because you can't start anywheres else.			
21.05			begin lateral movement	press-right-button	Display 0
21.70	22.34		SM begins jump/goal: stomp on goomba	press-A-button	Display 1
22.12			turns to face left in mid-flight at apogee	press-left-button	
22.06	23.53	Stomp on the GOOMBA and get			
	22.34		Stomp on goomba &		Display 2
22.34	23.51		begins second jump	press-A-button	
22.99			at apogee of second jump		
23.17			turns SM to face right	press-right-button	
	23.51		lands on ground &		
23.51	23.92		begins jump to CB1	press-A-button	
23.54	24.54	the coins.			
23.56			turns to face left	press-left-button	
23.67			hits CB1/begins to fall		

Transcript supplied by Wayne Gray				***Inferences made during our analysis***	
TIME (SEC.)				INFERRED	CORRESPONDING
START	END	VOICE	SUPER MARIO'S MOVEMENTS	FINGER ACTION	DISPLAY
23.71			turns to right	press-right-button	Display 3
	23.92		on ground &		
23.92	24.32		begins jump to CB2	press-A-button	
24.01			turns to left	press-left-button	
24.11			hits CB2/turns to right	press-right-button	Display 4
	24.32		on ground &		
24.32	24.79		begins jump	press-A-button	
24.64			turns to left	press-left-button	
	24.79		lands on pink block		
24.94	25.19		moves to left		
25.16	25.57	**After you get**			
25.19	25.76		at left edge of pink block, jumps	press-A-button	
25.27			turns to right	press-right-button	
25.39			hits CB3		
25.64	26.16	**the coins**			
25.71			turns to left	press-left-button	Display 5
	25.76		lands on ground (error?)		
25.91			turns to right	press-right-button	
25.99	26.54		jumps	press-A-button	
26.07			turns left	press-left-button	
26.18	26.65	**in the fourth**			
26.23			turns right	press-right-button	
26.51			turns left	press-left-button	
	26.54		lands on pink block (recovery?) &		
26.54	0.00		jumps	press-A-button	
26.65	27.05	**block**			
26.66			turns right	press-right-button	
26.73			hits Q-block		

Transcript supplied by Wayne Gray				*Inferences made during our analysis*	
TIME (SEC.)				INFERRED	CORRESPONDING
START	END	VOICE	SUPER MARIO'S MOVEMENTS	FINGER ACTION	DISPLAY
	26.93		lands on pink block		
26.99			turns left	press-left-button	Display 6
26.99	28.43		jumps (note that mushroom has not yet started to emerge from Q-block. This is an anticipation.)	press-A-button	
27.05	27.26	**get**			
27.53			gets mushroom @top of jump, becomes Super-Mario (shrinks & expands several times)		
28.12	28.78	**the aahh**			
28.28			turns right	press-right-button	Display 7
	28.43		lands on top of Q-block that mushroom emerged from.		
28.53	28.83		moves forward, off of Q-blk, begins fall down & to right		
28.78	29.35	**mushrooms**			
	28.83		lands on blue-block		
28.90	29.81		jumps	press-A-button	Display 8
29.35	31.31	**so you can turn into Super Mario**			
29.70			turns left	press-left-button	
	29.81		lands on green blk, beneath coin-blk, & to right of parana plant.		
29.85			turns right	press-right-button	Display 9
29.88	29.98		jumps	press-A-button	
29.93			hits coin-block		
	29.98		lands on green block (high green)		
30.20	31.13		jumps	press-A-button	Display 10
30.70			turns left -- floats while moving right	press-left-button	
31.06			turns right	press-right-button	
	31.13		lands on long green block, with KT on it.		

Transcript supplied by Wayne Gray				*Inferences made during our analysis*	
TIME (SEC.)				INFERRED	CORRESPONDING
START	END	VOICE	SUPER MARIO'S MOVEMENTS	FINGER ACTION	DISPLAY
31.14			jumps -- straight up	press-A-button	Display 11
31.38			at apogee begins to float to right	press-right-button	
31.56			turns left	press-left-button	
	31.70		stomps on KT (causing KT to retreat into its shell) &		
31.70	32.21		jumps towards the left	press-A-button	
31.73	31.98	And			
31.98	32.35	kick the			
	32.21		lands on long green blk &		
32.21			turns right	press-right-button	
32.35	32.85	koopa			
32.48			begins moving towards right (in the direction of the stomped upon KT)		
32.78			picks up KT	hold-B-button	
32.83			throws KT to right	release-B-button	
32.97	34.08		jumps towards right	press-A-button &	
				press-right-button	Display 12
33.08	34.31	so you can get the leaf			
33.12			thrown KT hits Q-blk, blk vanishes & returns		
33.20			leaf emerges from Q-blk & floats up.		
33.50			SM contacts leaf in mid-air, both vanish momentarily		
33.88			SM reappears as "Raccoon Mario" (R-M)		
	34.08		R-M lands on small blocks &		
34.08			turns left (sliding towards right)	press-left-button	Display 13
34.25	34.35		jump	press-A-button	
34.28			turns right	press-right-button	Display 14
34.28			hits coins box		
	34.35		lands		

Transcript supplied by Wayne Gray				*Inferences made during our analysis*	
TIME (SEC.)				INFERRED	CORRESPONDING
START	END	VOICE	SUPER MARIO'S MOVEMENTS	FINGER ACTION	DISPLAY
34.45			turns to face player (turns to front)		
34.52			turns left	press-left-button	Display 15
34.58			turns towards rear (with back towards player)		
34.65			turns to right	press-right-button	
34.73	35.28		jumps (goomba is coming)	press-A-button	
34.97			turns left	press-left-button	
35.02	36.04	**from getting the leaf**			
	35.28		stomps goomba &		
35.28			turns right	press-right-button	Display 17
35.28	36.10		jump	press-A-button	
35.75			turns left at apogee	press-left-button	
35.95			turns right	press-right-button	
	36.10		stomps another goomba &		
36.10			jumps	press-A-button (repeatedly)	Display 19
36.22	36.84	**go get**			
36.37			turns left (NOTE: flying KT is coming towards him) R-T flies higher and to the left	press-left-button	
36.50	37.02		turns right (floats in air)	press-right-button	
36.80			turns left (now over the flying KT who has now landed)	press-left-button	
36.89	38.43	**the other goomba so you can get**			
	37.02		stomps on flying KT (who loses his wings)		
37.02	37.85		jumps	press-A-button	Display 20
37.40			turns right at apogee	press-right-button	
37.51			turns left	press-left-button	
37.75			turns right (making many adjustments in order to land on the KT)	press-right-button	
	37.85		stomps KT		

Transcript supplied by Wayne Gray TIME (SEC.)				***Inferences made during our analysis***	
START	END	VOICE	SUPER MARIO'S MOVEMENTS	INFERRED FINGER ACTION	CORRESPONDING DISPLAY
37.85			turns left	press-left-button	Display 21
37.85	38.84		jumps	press-A-button	
38.43	38.63	more			
38.60			turns to rear		
38.65			turns left	press-left-button	
38.69	39.31	points			
	38.84		lands		
38.84			runs to left	press-left-button	
	39.52		stops (has gone to leaf blk, would have to jump to get by this)		
39.52			turns right	press-right-button	
39.52	41.87		runs to right, increasing speed	press-right-button &	Display 22 to Display 25
39.95	40.31	And then you		hold-B-button	
40.36	40.61	pick up			
40.61	40.88	enough			
41.03	42.46	running speed and fly		release-B-button &	Display 26
41.87	45.40		jumps -- flies	press-A-button (repeatedly)	
41.96			turns left	press-left-button	
42.11			gets first coin (in air)		
42.19			turns right (moving towards right)	press-right-button &	Display 27 to Display 30
42.46			gets coin	press-A-button (repeatedly)	
42.80			gets coin		
42.82			gets coin		
42.83	43.81	to get the coins			
43.19			gets coin		
43.56			gets coin		

Appendix II - Predictions of the GOMS Models

Display. 0 (Start-up screen) Mario starts at far left and Goomba is at far right moving left.
Mario is trying to get the first question block.

State of Mario: Small Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	far left, facing right				
	Ground1.1	all				
	QB.1	center, within reach	→	search-in-block(QB.1)	best better than search-in-block(QB.2) better than search-in-block(QB.3) better than search-in-block(QB.4)	search-in-block(QB.1)
	QB.2	just right of QB.1, within reach	→	search-in-block(QB.2)	best	
	QB.3	right of QB.2 too high	→	search-in-block(QB.3)	best	
	QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	
	G.1	far right (moving left)	→	attack-enemy(G.1)	worse than move-toward-end	
	Scaf2.1	under QB.4				
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	hit-from-bottom	move-horizontal (until under)	→	press-right-button	performed	<--- Triggers Next Display
		jump	→	press-A-button	not performed	

Display. 1 Goomba gets threateningly close to Mario, so Mario attacks it.

State of Mario: Small Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	center, facing right				
	Ground1.1	all				
	G.1	moving left, close to Mario	→	attack-enemy(G.1)	best	attack-enemy(G.1)
					better than search-in-block(QB.1)	
					better than search-in-block(QB.2)	
					better than search-in-block(QB.3)	
					better than search-in-block(QB.4)	
	QB.1	center, within reach	→	search-in-block(QB.1)	best	
	QB.2	just right of QB.1, within reach	→	search-in-block(QB.2)	best	
	QB.3	right of QB.2, too high	→	search-in-block(QB.3)	best	
	QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	
	Scaf2.1					
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	stomp-on	jump	→	press-A-button	performed	<--- Triggers Next Display

Display. 2 Mario goes on to get the closest question block (QB.1).

State of Mario: Small Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	center, facing right				
	Ground1.1	all				
	QB.1	center, within reach	→	search-in-block(QB.1)	best	search-in-block(QB.1)
					better than search-in-block(QB.2)	
					better than search-in-block(QB.3)	
					better than search-in-block(QB.4)	
	QB.2	just right of QB.1, within reach	→	search-in-block(QB.2)	best	
	QB.3	right of QB.2 too high	→	search-in-block(QB.3)	best	
	QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	
	Scaf2.1	under QB.4				
	Scaf3.2	just to the right of Scaf2.1				
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	hit-from-bottom	move-horizontal (until under)	→	press-right-button	performed	
		jump	→	press-A-button	performed	<--- Triggers Next Display

Display. 3 Mario goes on to get the next question block (QB.2).

State of Mario: Small Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	center, facing right				
	Ground1.1	all				
	QB.2	just right of QB.1, within reach	→	search-in-block(QB.2)	best	search-in-block(QB.2)
					better than search-in-block(QB.3)	
					better than search-in-block(QB.4)	
	QB.3	right of QB.2 too high	→	search-in-block(QB.3)	best	
	QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	
	Scaf2.1	under QB.4				
	Scaf3.2	just to the right of Scaf2.1				
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	hit-from-bottom	move-horizontal (until under)	→	press-right-button	performed	
		jump	→	press-A-button	performed	<--- Triggers Next Display

Display. 4 Mario goes on to get QB.3 but it is too high so he jumps onto a scafold to reach it.

State of Mario: Small Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	center				
Ground1.1	all				
QB.3	right of QB.2 too high	→	search-in-block(QB.3)	best better than search-in-block(QB.4)	search-in-block(QB.3)
QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	
Scaf2.1	under QB.4				
Scaf3.2	just to the right of Scaf2.1				
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED
hit-from-bottom	move-horizontal (until next to Scaf2.1)	→	press-right-button	performed
	jump (onto Scaf2.1)	→	press-A-button	performed
	move-horizontal (until under QB.3)	→	press-left-button	performed
	jump	→	press-A-button	performed

<--- Triggers Next Display

Display. 5 Mario jumps back onto the scafold to reach QB.4.

State of Mario: Small Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	on ground under QB.3, facing right				
Ground1.1	all				
QB.4	just right of QB.3, too high	→	search-in-block(QB.4)	best	search-in-block(QB.4)
Scaf2.1	under QB.4				
Scaf3.2	just to the right of Scaf2.1				
Plant.1	far right	→	avoid-danger (Plant.1)	worse than move-toward-end	
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED
hit-from-bottom	move-horizontal (until under)	→	press-A-button	performed
	jump (onto Scaf2.1)	→	press-right-button	performed
	jump (to QB.4)	→	press-A-button	performed

<--- Triggers Next Display

Display. 6 A Mushroom pops out of QB.4 so Mario chases it in order to become Super Mario.

State of Mario: Small Mario

Selection of F.L.O

OBJECTS SEEN	POSITION
Mario	on Scaf2.1, facing right
Ground1.1	all
Mush.1	above QB.4
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1
Plant.1	far right

EVOKES →

PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
gather-item(Mush.1)	best	gather-item(Mush.1)
avoid-danger (Plant.1)	worse than move-toward-end	
move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
run-into	move-horizontal (until at)	→	press-right-button	performed	<--- Triggers Next Display

Note: ***the Mushroom moves randomly to the left or right of the screen after popping out of QB.4. The situation in which it moves to the right was selected for this model.***

Display. 7 A plant is shooting Fireballs at Mario; he avoids them by running right up next to the plant.

State of Mario: Super Mario

Selection of F.L.O

OBJECTS SEEN	POSITION
Mario	on Ground1.1, left of Plant.1
Ground1.1	all
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1
Plant.1	right
Fireball	coming toward Mario
Scaf2.3	to the right of Plant.1

EVOKES →

PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
avoid-danger (Plant.1)	worse than move-toward-end	
avoid-danger(Fireball)	better than move-toward-end	avoid-danger (Fireball)
move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
get-out-of-the-way	move-horizontal (until out of range)	→	press-right-button	performed	<--- Triggers Next Display

Display. 8 Mario goes after the next question block by jumping over the plant and onto Scaf2.3.

State of Mario: Super Mario

Selection of F.L.O

OBJECTS SEEN	POSITION
Mario	on Ground1.1, next to Plant.1
Ground1.1	all
Plant.1	right (next to Mario)
Scaf2.3	right of Plant.1
QB.5	above Scaf2.3

EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
	avoid-danger (Plant.1)	worse than move-toward-end	
→	search-in-block(QB.5)	best	search-in-block(QB.5)
	move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-from-bottom	jump-horizontal (over Pipe.1)	→	press-A-button	performed	
		→	press-right-button	performed	
	jump-onto-Scaf2.3	→	press-A-button	performed	
		→	press-right-button	performed	
	move-horizontal-until-under	→	press-right-button	begun but not completed	<--- Triggers Next Display
	jump	→	press-A-button	not performed	

Display. 9 Goomba appears before Mario has reached QB.5 but it is not an immediate threat so Mario continues toward QB.5.

State of Mario: Super Mario

Selection of F.L.O

OBJECTS SEEN	POSITION
Mario	on Scaf2.3 under QB.5, facing right
Ground1.1	all
Plant.1	far left
Scaf2.3	right of Plant.1
QB.5	above Scaf2.3
G.5	on Ground1.1 in front of Scaf2.3

EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
→	avoid-danger (Plant.1)	worse than move-toward-end	
→	search-in-block(QB.5)	best	search-in-block(QB.5)
→	attack-enemy(G.5)	worse than move-toward-end	
	move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-from-bottom	move-horizontal-until-under	→	press-right-button	performed	
	jump	→	press-A-button	performed	<--- Triggers Next Display

Display. 10 Neither Goomba nor Koopa are an immediate threat.
Mario moves forward by jumping onto the scafolds.

State of Mario: Super Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	on Scaf2.3, facing right				
	Ground1.1	all				
	Scaf2.3	far left				
	Scaf3.4	center				
	Scaf4.5	right				
	Scaf1.6	below Scaf4.5				
	G.5	on Ground1.1 in front of scafolds	→	attack-enemy(G.5)	worse than move-toward-end	
	K.1	on Scaf1.6	→	attack enemy(K.1)	worse than move-toward-end	
				move-toward-end	worst	move-toward-end

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	move-right-and -jump	move-horizontal	→	press-right-button	performed	
		jump-onto-Scaf3.4	→	press-A-button	performed	
		move-horizontal	→	press-right-button	performed	
		jump-onto-Scaf4.5	→	press-A-button	performed	<--- Triggers Next Display

Display. 11 Mario sees block on the ground which can only be opened by thowing a shell at it. So Mario stomps on the Koopa Trooper in order to get a shell to throw at the block. Then Mario throws the shell at QB.6 and breaks it.

State of Mario: Super Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	on Scaf4.5, facing right				
Ground1.1	all				
Scaf2.3	far left				
Scaf3.4	center				
Scaf4.5	right				
Scaf1.6	below Scaf4.5				
G.5	on Ground1.1 in front of scafolds	→	attack-enemy(G.5)	worse than move-toward-end	
K.1	on Scaf1.6	→	attack enemy(K-1)	worse than move-toward-end	
Ground2.2	far right				
QB.6	far right on Ground2.2	→	search-in-block(QB.6)	best	search-in-block(QB.6)
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-with-shell	move-horizontal	→	press-right-button	performed	
	(drops onto Koopa to immobilize it)				
	prepare-to-pick-up	→	hold-B-button	performed	
	move-to-shell-and-pick-up	→	press-left-button	performed	
	move-to-block	→	press-right-button	performed	
	kick-shell	→	release-B-button	performed	
	jump (out of the way)	→	release-A-button	performed	<--- Triggers Next Display

***Note:** The last press-A-button is to jump out of the way of the richocheting Koopa shell. This KLO is added to the hit-with-shell method based on prior experience with being killed by the bouncing Koopa shell. See the text for more explanation of this learning process.*

Display. 12 Mario goes to get the leaf that popped out of the question block (QB.6) so that he can turn into Racoon Mario.

State of Mario: Super Mario

Selection of Operator

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	in air, facing right				
Ground1.1	left				
Scaf3.4	far left				
Scaf4.5	center				
Scaf1.6	below Scaf4.5				
G.5	on Ground1.1 in front of scafolds	→	attack-enemy(G.5)	worse than move-toward-end	
Ground2.2	right				
QB.6 (empty)	right, on Ground2.2				
Leaf.1	over QB.6	→	gather-item(Leaf.1)	best	gather-item(Leaf.1)
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
run-into	move-horizontal (until at)	→	press-right-button	performed	<--- Triggers Next Display

Display. 13 Mario goes to get the next question block (QB.7).

State of Mario: Racoon Mario

Selection of F.L.O

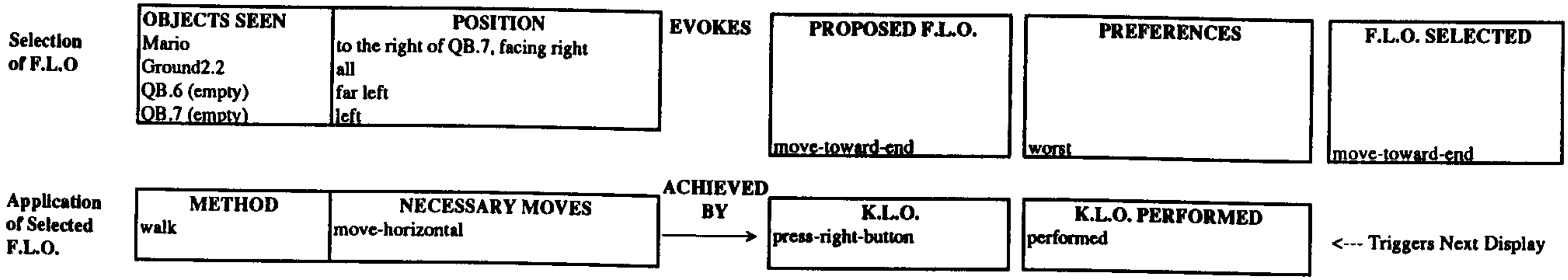
OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	between QB.6 and QB.7, facing right				
Ground1.1	left				
Scaf1.6	far left				
Ground2.2	right				
QB.6 (empty)	center, on Ground2.2				
QB.7	in air to the right of QB.6	→	search-in-block(QB.7)	best	search-in-block(QB.7)
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-from-bottom	move-horizontal (until under)	→	press-right-button	performed	
	jump	→	press-A-button	performed	<--- Triggers Next Display

Display. 14 There are no items or enemies on the screen so Mario just moves toward the end of the level.

State of Mario: Racoon Mario



Selection of F.L.O

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing right
Ground2.2	all
QB.6 (empty)	far left
QB.7 (empty)	left

EVOKES

PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
move-toward-end	worst	move-toward-end

Application of Selected F.L.O.

METHOD	NECESSARY MOVES
walk	move-horizontal

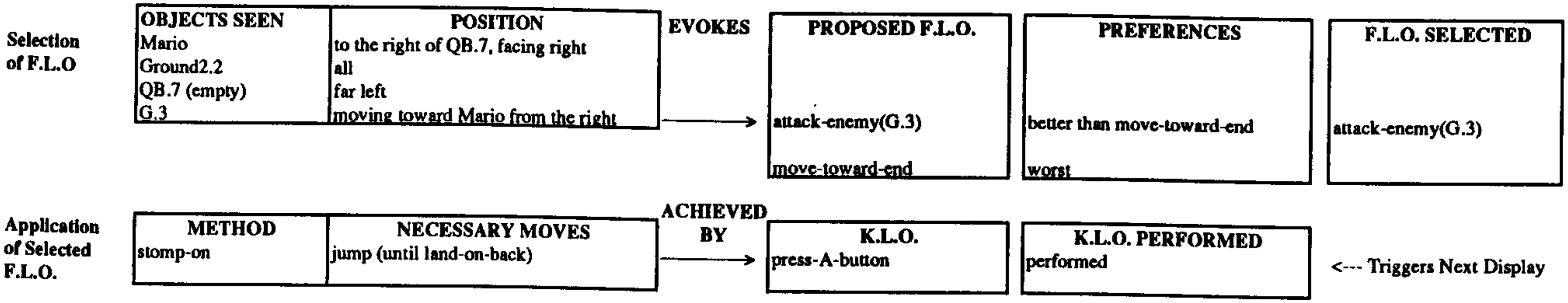
ACHIEVED BY

K.L.O.	K.L.O. PERFORMED
press-right-button	performed

<--- Triggers Next Display

Display. 15 A goomba (G.3) moves toward Mario from the right (becoming an immediate threat), so Mario attacks him.

State of Mario: Racoon Mario



Selection of F.L.O

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing right
Ground2.2	all
QB.7 (empty)	far left
G.3	moving toward Mario from the right

EVOKES

PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
attack-enemy(G.3)	better than move-toward-end	attack-enemy(G.3)
move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES
stomp-on	jump (until land-on-back)

ACHIEVED BY

K.L.O.	K.L.O. PERFORMED
press-A-button	performed

<--- Triggers Next Display

Display. 16 Again, there are no items or enemies on the screen so Mario just moves toward the end of the level.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the right of QB.7, facing right				
	Ground2.2	all		move-toward-end	worst	move-toward-end

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	walk	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display

Display. 17 Another goomba (G.4) moves toward Mario from the right (becoming an immediate threat), so Mario attacks him.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the right of QB.7, facing right				
	Ground2.2	all				
	G.4	moving toward Mario from the right	→	attack-enemy(G.4)	better than move-toward-end	attack-enemy(G.4)
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	stomp-on	jump (until land-on-back)	→	press-A-button	performed	<--- Triggers Next Display

Display. 18 Again, there are no items or enemies on the screen so Mario just moves toward the end of the level.

State of Mario: Racoon Mario

Selection of F.L.O	**OBJECTS SEEN**	**POSITION**	**EVOKES**	**PROPOSED F.L.O.**	**PREFERENCES**	**F.L.O. SELECTED**
	Mario	to the right of QB.7, facing right				
	Ground2.2	all		move-toward-end	worst	move-toward-end

Application of Selected F.L.O.	**METHOD**	**NECESSARY MOVES**	**ACHIEVED BY**	**K.L.O.**	**K.L.O. PERFORMED**	
	walk	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display

Display. 19 A para-goomba (PG.1) moves toward Mario from the right (becoming an immediate threat), so Mario attacks him.

State of Mario: Racoon Mario

Selection of F.L.O	**OBJECTS SEEN**	**POSITION**	**EVOKES**	**PROPOSED F.L.O.**	**PREFERENCES**	**F.L.O. SELECTED**
	Mario	to the right of QB.7, facing right				
	Ground2.2	all				
	PG.1	moving toward Mario from the right	→	attack-enemy(PG.1)	better than move-toward-end	attack-enemy(PG.1)
				move-toward-end	worst	

Application of Selected F.L.O.	**METHOD**	**NECESSARY MOVES**	**ACHIEVED BY**	**K.L.O.**	**K.L.O. PERFORMED**	
	stomp-on	jump (until land-on-back)	→	press-A-button	performed	<--- Triggers Next Display

Display. 20 The para-goomba turns into a goomba (G.5) after Mario stomps on it, so he has to attack it again.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the right of QB.7, facing right				
	Ground2.2	all				
	G.5	on Mario's left	→	attack-enemy(G.5)	better than move-toward-end	attack-enemy(G.5)
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	stomp-on	jump	→	press-A-button	performed	
		move-horizontal (until land-on-back)	→	press-left-button	performed	<--- Triggers Next Display

Display. 21 Again, there are no items or enemies on the screen so Mario just moves toward the end of the level.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the right of QB.7, facing right				
	Ground2.2	all		move-toward-end	worst	move-toward-end

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	walk	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display

Display. 22 A coin comes into view in mid-air at the far right; it is low enough for Mario to reach it by jumping.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the right of QB.7, facing right				
	Ground2.2	all				
	Coin.1	far right	→	gather-item(Coin.1)	best	gather-item(Coin.1)
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	hit-from-bottom	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display
		jump (when under)	→	press-A-button	not performed	

Display. 23 A cliff apears at the far right as Mario moves toward the coin.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the right of QB.7, facing right				
	Ground2.2	all				
	Coin.1	right, before Cliff.1	→	gather-item(Coin.1)	best	gather-item(Coin.1)
	Cliff.1	far right	→	avoid-danger(Cliff.1)	worse than move-toward-end	
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	hit-from-bottom	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display
		jump (when under)	→	press-A-button	not performed	

Display. 24 A second coin comes into view; this one is over the cliff and too high for Mario to jump.

State of Mario: Racoon Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	to the right of QB.7, facing right				
Ground2.2	all				
Coin.1	right, before Cliff.1	→	gather-item(Coin.1)	best	gather-item(Coin.1)
				better than gather-item(Coin.2)	
Cliff.1	right	→	avoid-danger(Cliff.1)	worse than move-toward-end	
Coin.2	far right over Cliff.1	→	gather-item(Coin.2)	best	
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-from-bottom	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display
	jump (when under)	→	press-A-button	not performed	

Display. 25 A third coin comes into view; this one is also over the cliff and too high for Mario to jump.

State of Mario: Racoon Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	to the left of Coin.1, facing right				
Ground2.2	all				
Coin.1	right, before Cliff.1	→	gather-item(Coin.1)	best	gather-item(Coin.1)
				better than gather-item(Coin.2)	
				better than gather-item(Coin.3)	
Cliff.1	right	→	avoid-danger(Cliff.1)	worse than move-toward-end	
Coin.2	far right over Cliff.1	→	gather-item(Coin.2)	best	
Coin.3	to the right and higher than Coin.2	→	gather-item(Coin.3)	best	
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
hit-from-bottom	move-horizontal	→	press-right-button	performed	<--- Triggers Next Display
	jump (when under)	→	press-A-button	not performed	

Display. 26 Mario arrives under Coin.1 and jumps to get it.

State of Mario: Racoon Mario

	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Selection of F.L.O	Mario	to the left of Coin.1, facing right				
	Ground2.2	all				
	Coin.1	right, before Cliff.1	→	gather-item(Coin.1)	best	gather-item(Coin.1)
					better than gather-item(Coin.2)	
					better than gather-item(Coin.3)	
	Cliff.1	right	→	avoid-danger(Cliff.1)	worse than move-toward-end	
	Coin.2	far right over Cliff.1	→	gather-item(Coin.2)	best	
	Coin.3	to the right and higher than Coin.2	→	gather-item(Coin.3)	best	
				move-toward-end	worst	

	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
Application of Selected F.L.O.	hit-from-bottom	move-horizontal (until-under)	→	press-right-button	performed	
		jump	→	press-A-button	performed	<--- Triggers Next Display

Display. 27 Mario cannot get the second coin by jumping to it so he has to back-up and fly to it.

State of Mario: Racoon Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	next to Cliff.1, facing right				
Ground2.2	all				
Cliff.1	right	→	avoid-danger(Cliff.1)	worse than move-toward-end	
Coin.2	over Cliff.1	→	gather-item(Coin.2)	best	gather-item(Coin.2)
				better than gather-item(Coin.3)	
Coin.3	to the right and higher than Coin.2	→	gather-item(Coin.3)	best	
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
fly-to-it	walk back	→	press-left-button	performed	
	turn around	→	press-right-button	performed	
	accelerate	→	press-right-button	performed	
		→	hold-B-button	performed	
	take off and fly	→	release-B-button	performed	
		→	press-A-button (repeatedly)	performed	<--- Triggers Next Display

Display. 28 After getting the second coin, Mario continues flying to the third coin, and a fourth and fifth coin come into view which he can also fly to.

State of Mario: Racoon Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	flying, facing right				
Coin.3	in the air	→	gather-item(Coin.3)	best better than gather-item(Coin.4) better than gather-item(Coin.5)	gather-item(Coin.3)
Coin.4	to the right and higher than Coin.3	→	gather-item(Coin.4)	best	
Coin.5	to the right and higher than Coin.4	→	gather-item(Coin.4)	best	
			move-toward-end	worst	

Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED
fly-to-it	fly	→	press-right-button	performed
		→	press-A-button (repeatedly)	performed

<--- Triggers Next Display

Display. 29 After getting the third coin, Mario continues flying to the fourth coin.

State of Mario: Racoon Mario

Selection of F.L.O

OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
Mario	flying, facing right				
Coin.4	in the air	→	gather-item(Coin.4)	best better than gather-item(Coin.5)	gather-item(Coin.4)
Coin.5	to the right and higher than Coin.4	→	gather-item(Coin.4)	best	
			move-toward-end	worst	

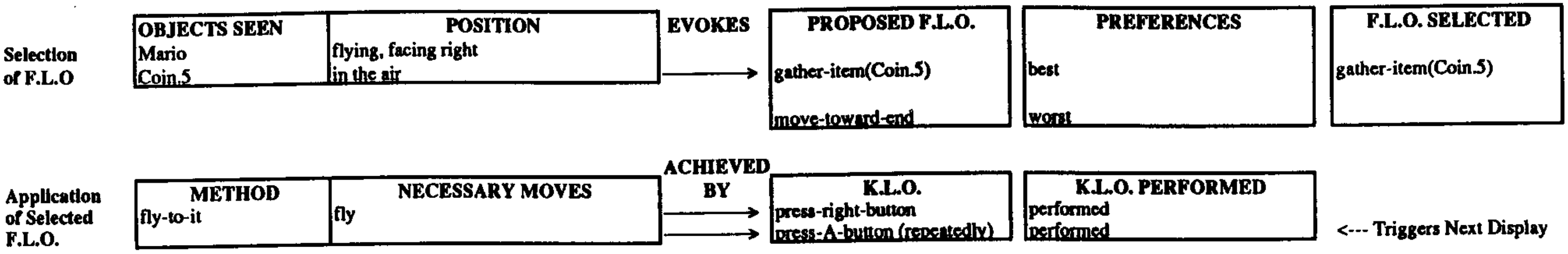
Application of Selected F.L.O.

METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED
fly-to-it	fly	→	press-right-button	performed
		→	press-A-button (repeatedly)	performed

<--- Triggers Next Display

Display. 30 After getting the fourth coin, Mario continues flying to the fifth coin, and a scafold appears with more coins above it.

State of Mario: Racoon Mario



Selection of F.L.O	OBJECTS SEEN	POSITION	EVOKES	PROPOSED F.L.O.	PREFERENCES	F.L.O. SELECTED
	Mario	flying, facing right	→	gather-item(Coin.5)	best	gather-item(Coin.5)
	Coin.5	in the air		move-toward-end	worst	

Application of Selected F.L.O.	METHOD	NECESSARY MOVES	ACHIEVED BY	K.L.O.	K.L.O. PERFORMED	
	fly-to-it	fly	→	press-right-button	performed	
			→	press-A-button (repeatedly)	performed	<--- Triggers Next Display

Appendix III - Comparison between Observed Expert Behavior and Predictions of the GOMS Models

Display. 0 (Start-up screen)

Mario starts at far left.
Goomba is at far right moving left.
Mario is trying to get the first question block.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	far left, facing right
Ground1.1	all
QB.1	center, within reach
QB.2	just right of QB.1, within reach
QB.3	right of QB.2 too high
QB.4	just right of QB.3, too high
G.1	far right (moving left)
Scaf2.1	under QB.4

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario moves right	ambiguous	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.1)	search-in-block(QB.1)	press-right-button
search-in-block(QB.2)		
search-in-block(QB.3)		
search-in-block(QB.4)		
attack-enemy(G.1)		
move-to-end		

Display. 1 Goomba gets threateningly close to Mario, so Mario attacks it.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	center, facing right
Ground1.1	all
G.1	moving left, close to Mario
QB.1	center, within reach
QB.2	just right of QB.1, within reach
QB.3	right of QB.2, too high
QB.4	just right of QB.3, too high
Scaf2.1	

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario jumps to the right		press-right-button
		press-A-button
turns left at apogee		press-left-button
lands on G.1	attack-enemy(G.1)	

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
attack-enemy(G.1)	attack-enemy(G.1)	press-A-button

Display. 2 Mario goes on to get the closest question block (QB.1).

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	center, facing left
Ground1.1	all
QB.1	center, within reach
QB.2	just right of QB.1, within reach
QB.3	right of QB.2 too high
QB.4	just right of QB.3, too high
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario jumps off G.1		press-A-button
turns to the right at apogee		press-right-button
jumps again		press-A-button
turns to left and hits QB.1	search-in-block(QB.1)	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.1)	search-in-block(QB.1)	press-right-button
		press-A-button

Display. 3 Mario goes on to get the next question block (QB.2)

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	center, facing left
Ground1.1	all
QB.2	just right of QB.1, within reach
QB.3	right of QB.2 too high
QB.4	just right of QB.3, too high
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario turns to right on ground		press-right-button
jumps		press-A-button
turns to left and hits QB.2	search-in-block(QB.2)	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.2)	search-in-block(QB.2)	press-right-button press-A-button

Display. 4 Mario goes on to get QB.3 but it is too high so he jumps onto a scafold to reach it.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	center, facing left
Ground1.1	all
QB.3	right of QB.2 too high
QB.4	just right of QB.3, too high
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario turns to right on ground		press-right-button
jumps		press-A-button
turns left and lands on Scaf2.1		press-left-button
jumps		press-A-button
turns to right and hits QB.3	search-in-block(QB.3)	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.3)	search-in-block(QB.3)	press-right-button
		press-A-button
		press-left-button
		press-A-button

Display. 5 From the scafold he can also reach QB.4.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on ground under QB.3, facing right
Ground1.1	all
QB.4	just right of QB.3, too high
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1
Plant.1	far right

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns to left and lands on ground		press-left-button
turns to right		press-right-button
jumps		press-A-button
turns left in air		press-left-button
turns right in air		press-right-button
turns left and lands on Scaf2.1		press-left-button
jumps		press-A-button
turns to right and hits QB.4	search-in-block(QB.4)	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.4)	search-in-block(QB.4)	press-right-button
		press-A-button (to Scaf2.1)
		press-A-button (hit QB.4)

Display. 6 Mario jumps up to get the mushroom which pops out from QB.6 before it even starts emerging from the question block.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on Scaf2.1, facing right
Ground1.1	all
Mush.1	above QB.4
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1
Plant.1	far right

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns left and lands on Scaf2.1		press-left-button
jumps to Mush.1	gather-item(Mush.1)	press-A-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item(Mush.1)	gather-item(Mush.1)	press-right-button (run into Mush.1)

Display. 7 Mario jumps from QB.4 onto Scaf3.2, avoiding the fireball.

State of Mario: Small Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on QB.4, facing left
Ground1.1	all
Scaf2.1	under QB.4
Scaf3.2	just to the right of Scaf2.1
Plant.1	right
Fireball	coming toward Mario
Scaf2.3	to the right of Plant.1

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
moves right landing on Scaf3.2	avoid-danger(Fireball)	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
avoid-danger(Fireball) move-toward-end	avoid-danger(Fireball)	press-right-button

Display. 8 Mario goes after the next question block by jumping over the plant and onto Scaf2.3.

State of Mario: Super Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on Scaf3.2, facing right
Ground1.1	all
Plant.1	right
Scaf2.3	right of Plant.1
QB.5	above Scaf2.3

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario jumps over plant		press-A-button
turns left and lands on Scaf2.3	ambiguous	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.5)	search-in-block(QB.5)	press-A-button
avoid-danger(Plant.1)		press-right-button
move-toward-end		press-A-button
		press-right-button
		press-right-button

Display. 9 Goomba appears before Mario has reached QB.5.
It is not an immediate threat so Mario continues toward QB.5.

State of Mario: Super Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on Scaf2.3 under QB.5, facing left
Ground1.1	all
Plant.1	far left
Scaf2.3	right of Plant.1
QB.5	above Scaf2.3
G.5	on Ground1.1 in front of Scaf2.3

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns to the right		press-right-button
jumps to QB.5	search-in-block(QB.5)	press-A-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.5)	search-in-block(QB.5)	press-right-button press-A-button

Display. 10 Neither Goomba nor Koopa are an immediate threat and Mario drops down onto Scaf1.6.

State of Mario: Super Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on Scaf2.3, facing right
Ground1.1	all
Scaf2.3	far left
Scaf3.4	center
Scaf4.5	right
Scaf1.6	below Scaf4.5
G.5	on Ground1.1 in front of scafolds
K.1	on Scaf1.6

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
jumps		press-A-button
turns left		press-left-button
turns right and lands on Scaf1.6	ambiguous	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
attack-enemy(K.1) move-toward-end	move-toward-end	press-right-button press-A-button press-right-button press-A-button

Display. 11 Mario sees QB.6 on the ground which can only be opened by thowing a shell at it. So Mario stomps on the Koopa Trooper in order to get a shell to throw at the block. Then Mario throws the shell at QB.6 and breaks it.

State of Mario: Super Mario

On Screen

OBJECTS SEEN	POSITION
Mario	on Scaf1.6, facing right
Ground1.1	all
Scaf2.3	far left
Scaf3.4	center
Scaf4.5	right
Scaf1.6	below Scaf4.5
G.5	on Ground1.1 in front of scafolds
K.1	on Scaf1.6
QB.6	far right on Ground2.2

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
jumps straight up		press-A-button
moves right in air		press-right-button
turns left and lands on K.1		press-left-button
jumps, landing left of the shell		press-A-button
turns right		press-right-button
picks up Koopa shell		hold-B-button
kicks Shell at QB.6		release-B-button
jumps out of way of Koopa shell	search-in-block(QB.6)	release-A-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.6)	search-in-block(QB.6)	press-right-button hold-B-button press-left-button press-right-button release-B-button press-A-button

***Note:** The last press-A-button is to jump out of the way of the richocheting Koopa shell. This KLO is added to the hit-with-shell method based on prior experience with being killed by the bouncing Koopa shell. See the text for more explanation of this learning process.*

Display. 12 A leaf pops out of the QB.6.
Mario goes to get it so he can turn into Racoon Mario.

State of Mario: Super Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in air, facing right
Ground1.1	left
Scaf3.4	far left
Scaf4.5	center
Scaf1.6	below Scaf4.5
G.5	on Ground1.1 in front of scafolds
Ground2.2	right
QB.6 (empty)	right, on Ground2.2
Leaf.1	over QB.6

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
Mario moves to the right	gather-item(Leaf.1)	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item(Leaf.1)	gather-item(Leaf.1)	press-right-button

Display. 13 Mario goes to get the next question block (QB.7)

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	between QB.6 and QB.7, facing right
Ground1.1	left
Scaf1.6	far left
Ground2.2	right
QB.6 (empty)	center, on Ground2.2
QB.7	in air to the right of QB.6

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
moves right to QB.7		press-right-button
jumps		press-A-button
turns left and hits QB.7	search-in-block(QB.7)	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
search-in-block(QB.7)	search-in-block(QB.7)	press-right-button press-A-button

Display. 14 There are no items or enemies on the screen so Mario just moves toward the end of the level.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing left
Ground2.2	all
QB.6 (empty)	far left
QB.7 (empty)	left

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
moves right	move-toward-end	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
move-toward-end	move-toward-end	press-right-button

Display. 15 A goomba (G.3) moves toward Mario from the right (becoming an immediate threat), so Mario attacks him.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing right
Ground2.2	all
QB.7 (empty)	far left
G.3	moving toward Mario from the right

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns left		press-left-button
turns right		press-right-button
jumps		press-A-button
turns left and lands on G.3	attack-enemy(G.3)	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
attack-enemy(G.3)	attack-enemy(G.3)	press-A-button

Display. 16

This is not a distinct display in the expert's protocol.
Since the expert has moved away from QB.7 instead of waiting for G.3 to approach (as the model does), G.4 appears on the screen before the expert has stomped G.3.

Display. 17

Another goomba (G.4) moves toward Mario from the right (becoming an immediate threat), so Mario attacks him.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing left
Ground2.2	all
G.4	moving toward Mario from the right

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns right		press-right-button
jumps		press-A-button
turns left at apogee		press-left-button
turns right and lands on G.4	attack-enemy(G.4)	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
attack-enemy(G.4)	attack-enemy(G.4)	press-A-button

Display. 18 This is not a distinct display in the expert's protocol.
Since the expert has continued to move right instead of waiting for enemies to approach (as the model does), PG.1 appears on the screen before the expert has stomped G.4.

Display. 19 A para-goomba (PG.1) moves toward Mario from the right (becoming an immediate threat), so Mario attacks him.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing right
Ground2.2	all
PG.1	moving toward Mario from the right

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns left		press-left-button
jumps (higher than flying PG.1)		press-A-button (repeatedly)
turns right		press-right-button
turns left and lands on PG.1	attack-enemy(PG.1)	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
attack-enemy(PG.1)	attack-enemy(PG.1)	press-A-button

Display. 20 The paragoomba turns into a goomba (G.5) after Mario stomps on it, so he attacks it again.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	to the right of QB.7, facing left
Ground2.2	all
G.5	on Mario's left

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
jumps		press-A-button
turns right at apogee		press-right-button
turns left		press-left-button
turns right and lands on G.5	attack-enemy(G.5)	press-right-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
attack-enemy(G.5)	attack-enemy(G.5)	press-A-button
		press-left-button

Display. 21 Mario runs back to QB.6(empty) and then runs to the right accelerating to fly.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing right
Ground2.2	all

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns left		press-left-button
jumps		press-A-button
lands and runs to left		press-left-button
turns right		press-right-button
runs to right		press-right-button
accelerating	ambiguous	hold-B-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
move-toward-end	move-toward-end	press-right-button

Note: The first three observed behaviors (where Mario turns left, jumps, and then runs to the left) are not consistent with any proposed F.L.O.s. The other three behaviors (where Mario turns right and then runs to the right while accelerating) are consistent with move-toward-end. The whole sequence of behavior is both ambiguous and consistent with move-toward-end.

Display. 22 The first coin comes into view as Mario is picking up speed to fly.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing right
Ground2.2	all
Coin.1	next to Cliff.1

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
runs to right		press-right-button
accelerating	ambiguous	hold-B-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.1)	gather-item (Coin.1)	press-right-button
move-toward-end		

Display. 23 A cliff apears at the far right as Mario moves toward the coin.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing right
Ground2.2	all
Coin.1	next to Cliff.1
Cliff.1	far right

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
runs to right		press-right-button
accelerating	ambiguous	hold-B-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.1)	gather-item (Coin.1)	press-right-button
move-toward-end		

Display. 24 A second coin comes into view; this one is over the cliff and too high for Mario to jump.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing right
Ground2.2	all
Coin.1	next to Cliff.1
Cliff.1	right
Coin.2	far right and above Coin.1

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
runs to right		press-right-button
accelerating	ambiguous	hold-B-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.1)	gather-item (Coin.1)	press-right-button
gather-item (Coin.2)		
move-toward-end		

Display. 25 A third coin comes into view; this one is also over the cliff and too high for Mario to jump.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing right
Ground2.2	all
Coin.1	next to Cliff.1
Cliff.1	right
Coin.2	right and above Coin.1
Coin.3	far right and above Coin.2

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
runs to right accelerating	ambiguous	press-right-button hold-B-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.1) gather-item (Coin.2) gather-item (Coin.3) move-toward-end	gather-item (Coin.1)	press-right-button

Display. 26 Mario begins flying up to Coin.1.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing right
Ground2.2	all
Coin.1	next to Cliff.1
Cliff.1	right
Coin.2	over Cliff.1
Coin.3	to the right and above Coin.2

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
has enough speed to fly		release-B-button
jumps and flies		press-A-button (repeatedly)
turns left and gets first coin	gather-item (Coin.1)	press-left-button

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.1)	gather-item (Coin.1)	press-right-button
		press-A-button

Display. 27 Mario continues flying to the second coin.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	in center on Ground2.2, facing left
Ground2.2	all
Cliff.1	right
Coin.2	over Cliff.1
Coin.3	to the right and above Coin.2

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
turns right and flies to next coin	gather-item (Coin.2)	press-right-button press-A-button (repeatedly)

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.2)	gather-item (Coin.2)	press-left-button press-right-button press-right-button hold-B-button release-B-button press-A-button (repeatedly)

Display. 28 After getting the second coin, Mario continues flying to the third coin. Two more coins appear above and to the right of the third one.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	flying, facing right
Coin.3	in the air
Coin.4	to the right and above Coin.3
Coin.5	to the right and above Coin.4

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
flies to next coin	gather-item (Coin.3)	press-right-button press-A-button (repeatedly)

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.3)	gather-item (Coin.3)	press-right-button press-A-button (repeatedly)

Display. 29 After getting the third coin, Mario continues flying to the fourth coin.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	flying, facing right
Coin.4	in the air
Coin.5	to the right and above Coin.4

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
flies to next coin	gather-item (Coin.4)	press-right-button press-A-button (repeatedly)

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.4)	gather-item (Coin.4)	press-right-button press-A-button (repeatedly)

Display. 30 After getting the fourth coin, Mario continues flying to the fifth coin.

State of Mario: Racoon Mario

On Screen

OBJECTS SEEN	POSITION
Mario	flying, facing right
Coin.5	to the right and above Coin.4

Expert's Behavior

OBSERVED BEHAVIOR	INFERRED FUNCTION	INFERRED FINGER ACTION
flies to next coin	gather-item (Coin.5)	press-right-button press-A-button (repeatedly)

Comparison to Model

F.L.O. CONSISTENT WITH OBSERVED BEHAVIOR	F.L.O. PREDICTED BY MODEL	K.L.O. PREDICTED BY MODEL
gather-item (Coin.5)	gather-item (Coin.5)	press-right-button press-A-button (repeatedly)