92-36

Machine-Mediated Learning, Volume 3, pp. 81-105
Printed in the UK.

0732-6718/89 $3.00 + .00

Semantically Constrained Exploration and Heuristic Guidance

HENRY HAMBURGER
AKHTAR LODGHER

George Mason University
Fairfax, VA

Abstract *Exploration can be an effective learning experience, if suitably constrained and guided. Moreover, it can provide this benefit for specifically targeted formal skills in the arithmetic curriculum. This paper presents two complementary techniques for promoting success in computer-based exploration environments. Semantic constraints on eploration cut out meaningless options, and heuristic guidance facilitates search on the basis of a heuristic function with both cognitive and problem-solving components. We have implemented an environment that permits semantically constrained exploration for subtraction, as well as a related environment that facilitates the transition to paper-and-pencil subtraction. The authors tested the system in individual hour-long sessions with more than twenty children in grades 1-3.*

Keywords Exploration, heuristic, intelligent, arithmetic, problem-solving, prerequisite, transfer, coins.

Introduction

Exploration that leads to discovery can be a motivating and effective learning experience, but exploration can also degenerate into groping, leading to frustration and misconception. These twin observations, or variations on them, have long been present in the teaching literature; see [1, pp. 73–75] for discussion and references in the realm of arithmetic. This paper presents two complementary techniques for promoting success in computer-based exploration environments: semantically constrained exploration (SCE) and heuristic guidance (HG). The constraints in SCE are intended to cut out meaningless options that would only be distractions from useful exploration. HG further facilitates the learner's search by offering warnings and hints on the basis of a heuristic function with both cognitive and problem-solving components. The strategy is to provide useful individual attention without speculating excessively on the student's exact thoughts. We have implemented an environment for SCE in the domain of subtraction and are extending it to provide HG. The first two sections of the paper introduce and discuss SCE and HG. Each presentation makes extended reference to subtraction, for which some domain analysis appears in the third section.

Semantically Constrained Exploration

We seek to create an environment in which search will most often be gratifyingly successful and educationally beneficial. Semantic constraints are our first strategic move in the pursuit of that objective. The imposition of constraints has the effect of prepruning the search space and abstractly underlies the learner's possible actions in the environment. The trick is to make the constraints tight enough so that the search space is of proportions manageable for the target population of learners, yet not so tight as to undermine the interest and educational value of the activity.

The constraints we use have the additional property of being semantic in that they have a reasonable interpretation in the story or description of the environment and therefore make immediate sense to a learner. This claim is related to the idea of conceptual fidelity [2], and is elaborated below. In sum, the rationale for SCE is to facilitate hypothesis formation (active thought) by making good hypotheses readily available, and yet to relieve the learner of the cognitive load of memorizing seemingly arbitrary strictures. In the first three subsections, we introduce a particular SCE, justify it pedagogically, and relate it to other techniques. The fourth subsection is on representation, and the final one treats the transfer of skills to traditional representation.

Coinland: An Example of an SCE

To fix ideas about what an environment for SCE can look like, we provide a sketch of a particular one, Coinland, which we introduced in [3]. Coinland provides an environment for exploring elementary arithmetic operations. By doing so, it supports active problem-solving in a domain, arithmetic, in which many children can succeed at problem-solving [4]. At the same time, it enforces and thereby facilitates the learning of specific arithmetic principles, namely, place value representation and the conservation of quantity. Coinland is accompanied by an alternative but related representation, Numberland, which embodies the same concepts but has a more numerical aspect. Numberland moves the learner closer to the traditional world of pencil-and-paper arithmetic performance.

To set the stage for Coinland, let us introduce an apprentice in a fictional Coinland. This project may then be seen as providing a simulated apprenticeship [5]. Our Coinland is real (implemented) but imaginary (made of screen images). Imagine instead a Coinland that is imaginary (fictional) but real (made of physical entities, not computed ones). This fictional Coinland is a society without writing, in which people exchange goods and three types of coins, called Is, Xs and Cs. The different value levels of the coins give rise to conventions about exchanging them: ten Is for an X and ten Xs for a C.

Now, in this society there live an artisan and her son, the apprentice. The artisan has several product types, each with a price, and she keeps on hand a supply of coins, for making change. She is training her son to handle transactions in her shop, so that one day she can go off and do materials research. Lacking writing, she records the various prices by leaving a set of coins with each product type. Lacking (abstract) arithmetic, she tells him to extract the price of an object from whatever coins are proffered by a customer, having first executed coin exchanges as needed to make such extraction possible. Is this instruction sufficient? If not, what more does the apprentice need? When these folks make change correctly, do we credit them with knowing subtraction? And if

not, what more is there to it? We seek answers to these questions in our computational Coinland (henceforth simply "Coinland"), to which we now turn.

The student in a Coinland subtraction problem plays the role of a cashier in a store. A customer buys an item for some given price, selected by the system. The price, which is the subtrahend (the number to be subtracted), appears on the screen in numerical form. The money tendered corresponds to the minuend (the number from which the subtrahend is subtracted). This number is also picked by the system and takes the form of coins, shown as color-coded circles (yellow for pennies, etc.). The student's task is to give the correct change, keeping a set of coins identical to the price, thereby subtracting the price from the initially tendered amount. A change-making machine is available. For simplicity here, we shall use only pennies, dimes, and dollars, which suffice to introduce base-10 subtraction of three-digit numbers with borrowing across zero, as well as easier problems with one- or two-digit numbers. Varying the set of denominations gives an even broader range of problems, supporting the learning of additional concepts. Within the existing framework one could introduce a thousands column and/or a decimal column, or even move to base-5 arithmetic, with pennies, nickels and quarters.

The Coinland subtraction screen appears in Figure 1, which permits a more detailed description of the task. The particular situation shown is partly through the solution of the problem 861 − 307. Initially, the minuend, which is the money from the customer, was represented in the top row in the form of coins, in this case 8 dollars, 6 dimes and a penny. However, in the figure the student has already exchanged a dime for ten pennies to get a total of 11 pennies in the take-in tray, and then lowered 4 of them to the give-back tray. Notice that during solution, the student no longer has available a statement of the problem. Although that information is not necessary for the task, an earlier version of Coinland did keep it on the screen in numerical form, but it seemed to interfere more than help. In the second row appears the price of the item, expressed as a number, which serves as the subtrahend. The student's task is to get the correct amount of change into the "give-back" tray in the bottom row. This row starts out empty and should end up holding a representation in coin form of the answer to the subtraction problem (the tendered amount minus the price). The student can tell whether the problem has been solved correctly, since if it has, the take-in tray will hold a set of coins equal in value to the price.

Each of the two trays has three compartments for the three denominations. The student can manipulate the situation in two general ways, as exemplified above: (1) by shifting coins from one place to another, and (2) by converting between adjacent denominations. Shifts can be in either direction, from the take-in tray to the give-back tray, or, if necessary, from there to the take-in tray. A single shift can involve any number of coins, provided they all come from a single compartment of one tray (and are therefore of the same denomination) and go to the corresponding compartment of the other tray.

The learner navigates around the screen with the four arrow keys, picks up and drops coins with three specially labeled function keys, and uses the ENTER key to select options on the screen. (We also have an implementation in which a mouse controls movement and the mouse button triggers all actions). To shift coins, a user must (1) move the cursor, which plays the role of a hand, to the intended source compartment (using the arrow keys), (2) press the "PICK ONE COIN" key once per coin to be picked up, (3) move to the destination, and (4) press the "DROP ALL COINS" key. When a coin is picked up, it turns white at its source until it is dropped at its destination. One might say the white coin connotes a ghost or trace of the picked coin. The ghost

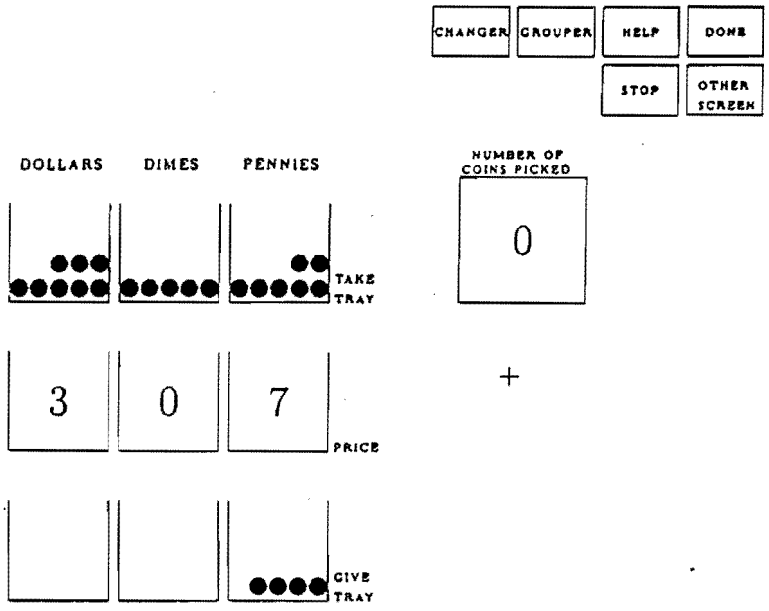

Figure 1 The Coinland screen during solution of the problem to give change from 861 when the price is 307. The student has exchanged one dime, leaving 5 of the original 6 dimes, and obtaining 10 extra pennies for a total of 11. Four of the 11 pennies have been lowered to the "give-back" tray and 7 remain in the "take-in" tray.

disappears when that coin is dropped at its destination. The number of coins picked up but not yet dropped appears on the screen as a numeral in a special box. There is a "DROP ONE COIN" key which can be handy for adjustment when you pick up too many coins. Dropping one coin back to its source restores color to a ghost coin.

Also on the screen are a changer and grouper. The changer converts a dime into ten pennies or a dollar into ten dimes. To use it, one simply clicks on it after having picked up a permissible input. The output coins go automatically into the appropriate compartment of the tray its input coin came from. The operations of the grouper are precisely inverse to those of the changer. Optimal plans for subtraction eschew the grouper, but it may be needed by some learners to undo unnecessary use of the changer. Also, the grouper is crucial earlier in the arithmetic curriculum for addition problems as well as for the teaching of place value, and we plan to use it later with multiplication.

Another important feature of our environment is the provision of an alternative representation. The student can move back and forth at will between Coinland and a Numberland with the same conceptual underpinnings, carrying out an equivalent search. Facility with the translation process carries the learner close to the ultimate transition, transfer of knowledge from the computer screen to the world of pencil and paper. This step is important, since "strong semantic understanding . . . in no way guarantees ability to use [a corresponding algorithm]" [6]. The grand strategy is that the student combines learning in Coinland, where things make sense, with the understanding that everything in Numberland has its equivalent in Coinland, to achieve not only correct performance in Numberland, but also the conviction that there too everything makes sense. A fuller discussion of Numberland, especially its role in promoting the transfer of skills, is the topic of the last subsection of this section.

Pedagogical Significance

This section highlights and justifies some design decisions, first concerning the interface and then the nature of the constraints on exploration. A screen world such as Coinland establishes conventions about the appearance of objects and the operations on them. For the student to readily understand, retain, and use such conventions, they should map accurately and consistently to simple familiar objects in the real world. What do children in the early grades know about the world that the conventions of Coinland draw upon? At the simplest level, children from before the age of one year exhibit a sense of object permanence, an expectation that objects, although they can be moved, do not just appear or disappear without reason. In Coinland, there is object permanence except when the student activates the grouper or the changer, and even then, the conversion process is emphasized by animation of the coins entering and leaving. For example, if a dime goes into the changer, ten pennies are shown emerging one after the other, accompanied by sound effects for emphasis. The explosion of common nouns in the vocabulary of two-year-olds is one form of evidence that youngsters realize that there are types or categories of objects, so that things that look the same tend to behave the same. Coinland coins of the same denomination are indistinguishable in appearance and behavior. Children also know, before school age, about the notions of ownership, keeping, giving, and buying that our scenario draws upon.

A specifically arithmetic notion for which a convention must be established is magnitude correspondence, the idea that larger numbers correspond to larger amounts of something. Dienes blocks gives a complete correspondence of volumes to numerical magnitudes, whereas Coinland provides a similar correspondence within denominations but not across them, since coins of different denominations are distinguished only by color, not by size. Across denominations we draw instead on conventions established by the money system, with which most of the children are familiar [7]. The changer and the grouper reify the exchange operations, giving a realistic procedural interpretation of the equivalence between one dime and ten pennies, and between one dollar and ten dimes. The changer and grouper conserve monetary quantity in Coinland, and they conserve numerical quantity in Numberland. Admittedly, conservation of quantity is a less transparent notion than object permanence, but children do typically encounter and master coin exchange, not to mention coin-operated machines, before they reach formal multidigit subtraction.

A final word on the pedagogy of the interface is in order. A major and robust finding in psychology is the efficacy of active as opposed to passive learning. For this reason, we have elected to have the student actively operate upon the set of coins that end up constituting the answer. We rejected an alternative arrangement in which the answer to a problem would have emerged passively, comprising the remaining coins (the "remainder") after removal of an amount equal to the price.

Let us now move past (or through!) the interface to explore the benefits of constrained exploration. Conservation of quantity constrains the search space, making it manageable and instructive. Perhaps most importantly, the conservation property permits circumvention of documented types of misperformance in subtraction [8]. Take the error of borrowing directly from the hundreds to the ones column. This action simply cannot be expressed in our representation. In other words, the privilege of making this error is an option that the student unwittingly relinquishes by acclimating to the Coinland environment.

A few examples will further illustrate the potential benefits of the semantic con-

straints. The conservation of quantity has as its simplest role the gentle enforcement of one-digit subtraction facts. In the problem 8 − 3, suppose the learner begins by moving, say, 6 pennies to the give-back tray instead of 5, in effect, trying to break 8 into 3 and 6. This is an attempted violation of the conservation of quantity, which the system does not support, in the sense that only 2 pennies (not 3) will be left in the take-in tray. The system provides the same type of enforcement of intra-column subtraction facts like 13 − 4 = 9 whose need arises after borrowing.

Consider now the role of semantic constraints in the more challenging example of 100 − 7. This problem requires subtraction from 0 and borrowing across 0, both of them operations known to cause difficulty. In the Coinland version of this problem, the take-in tray has a dollar at the outset and must in the goal state hold 7 pennies and nothing else. What can the student do to achieve such an outcome? Since the dollar is the only object available at the outset, there are only two possible moves: shift the dollar to the give-back gray or exchange it for ten dimes. Of course, one hopes the student will realize that getting dimes in this way satisfies a precondition for getting the needed pennies. The student will have learned this way of getting pennies earlier in the curriculum from two-digit subtraction problems and from preliminary exercises with the changer.

We hold that such problem solving has a much better change to emerge in the constrained search space of our SCE environment than in the relatively bushy search space of unconstrained paper-and-pencil subtraction that gave rise to the repair theory of bugs [9]. (A bushy space is one that tends to have many options at a point.) In the pencil world, there are many operations, and their various conditions are not easy to remember or determine. In Coinland there are very few operations and the tests for their legal applicability correspond closely to what would be physically possible with real coins. For example, if you haven't got something, say, a dime as in the preceding example, then you just can't do anything with it or to it.

Technique

To put the technique of SCE into perspective, we compare it to related techniques and consider some variations. Exploration of an environment is a learning mode that is characterized by the fairly high degree of control that the student is permitted to exercise, in comparison to other learning modes or educational techniques. On the spectrum from relatively free exploration to drill and practice, Coinland occupies a middle position, allowing the learner to control some aspects of the situation but not others. Specifically, our learner is free to choose moves within a problem, even in violation of the tutor's suggestions, but it is (currently) the tutor that chooses the problem, with a view to progressing in a specific syllabus. With many other techniques there is substantially greater reliance on the direct presentation of material. That presentation may take place via language, either in written form or (ultimately) spoken, and either undirectionally or interactively via questions, with the learner doing either the asking or the answering. As a complement or an alternative to language, material can be presented spatially, with diagrams, animation, pictures, or motion pictures.

Within the particular realm of exploration environments, one can distinguish at least three flavors, according to whether the concepts to be explored constitute a knowledge base, a programming language, or a problem in some domain such as subtraction. The latter two, programs and problems, provide an interesting comparison, as can be seen by looking at some important similarities and differences in the exploration of Coinland and

of Papert's world of turtle geometry [10]. First, the turtle's primitive actions of walking, turning, and drawing certainly conform to the above demand for an accurate mapping from an artificial learning environment into familiar aspects of the real world. A further similarity is that with both the turtle and the coins, the learner's task is arguably to assemble primitive operations into an algorithm that solves a problem. The crucial difference is that programming the turtle requires an explicit statement of the looping and branching for a particular problem, whereas in Coinland, as in paper-and-pencil arithmetic, one needs only implicit knowledge equivalent to a correct algorithm. The Coinland learner need not be able to state a correct algorithm to be deemed correct, just be able to act as if she were executing one. (An analogous case arises in speaking a language, where one need not be able to state the grammatical rules, just produce sentences governed by them).

To consider more explicitly the issue of explicitness, suppose we shared Papert's worthy aim of promoting explicit exploration of algorithms. Then perhaps we should have invented a Coinland programming language. About the simplest form such a language could take is one that allows one-digit subtraction to include code something like that shown just below. Although a child might work successfully at this level of complexity, the move to multidigit numbers would require a more complex language, specifying columns. An explicit procedural solution encompassing borrowing is demanding indeed, since it requires nested loops. (On the difficulty for preschoolers of even implicit handling of nested loops, see [11].)

```
GO-TO                TOP;
DO                   PICK-UP-ONE
UNTIL                TOP = BOTTOM;
GO-TO                ANSWER;
RELEASE;
```

Methods involving props like coins and Dienes blocks have been used without computers for a long time [12]. The computer adds flexibility in the creation of new environments and variations in them (as well as a potential for individual attention that may otherwise be in short supply from human teachers). An example of this flexibility is the choice of whether or not to permit the commingling of coins of different denominations. In the current version of Coinland, there simply is no mechanism for putting down coins in a compartment not of their own denomination. It would be easy enough to change the implementation to let the student choose an arbitrary target location, but we have chosen not to, since that would expand the search space. This design decision is easy to implement computationally, but in traditional setting with physical coins the prohibition of commingling could only be expressed as a rule that the student would have to remember, thereby potentially interfering with concentration on important conceptual matters.

Another design decision that the computer facilitates involve a technique introduced below, the use of partially covered compartments in the answer tray, to discourage removing coins from the answer once they are put there. The computer also deftly accommodates large numbers of coins, no small consideration in view of the observation that "teachers usually stop using objects to represent mathematical processes in second or third grade, when the numbers get bigger, [and] the objects become more unwieldy" [7].

To take a final viewpoint for comparison, regard Coinland as a one-person game. In conformity with Malone's [13] important observations about games, Coinland has no

conceptually wrong moves, just inefficient ones. In this context, taking more moves than necessary to reach a correct answer means nothing worse than achieving a less than maximum score. Consequently, there need be no inhibitions about trying things out. A recent survey of some relevant game-like techniques is in [14].

Representation

Good representation can help make ideas clear or even obvious by facilitating their mapping to analogous subject matter that is well understood. As noted above, good mapping is one desideratum we claim for the Coinland representation; another is versatility. Versatility is desirable for a representation, because it means that once the conventions have been learned, they can be used for the learning of a broad range of subject matter. The applicability of Coinland transcends subtraction, clearly extending downward in the arithmetic curriculum to addition, counting forward and backward and finding the larger of two numbers. Moreover, we intend to use most of its representational conventions for more advanced subject matter, including multiplication, division by very small integers, the use of a radix other than 10, and the addition and subtraction of small mixed numbers. We have hopes that Coinland will make it possible to teach beginning division in a particularly meaningful way, independently of multiplication, relying directly on an experience that many children have been exposed to: dividing things to share them.

The use of objects in the representation, in place of numerals, is a move toward concreteness, versus abstractness. It also entails more entities and actions, be they abstract or concrete, and in this sense it is relatively detailed. For example, in place of the one symbol "5," there are five objects, and the solution process for "8 − 5" involves three operations of picking up, rather than accessing the single atomic fact that 8 − 5 = 3. By failing to afford a way to use the atomic fact, the representation makes the learner show the validity of that fact in the course of using it. Erroneous beliefs about such facts can be discovered in this way.

In this regard, our approach bears an interesting relationship to that of White and Frederiksen [15] in the domain of electrical circuits. In their system, the learner starts with a qualitative model that moves away from quantitative detail. One of their rationales is that the simplicity of this representation makes it suitable for beginners. Teaching a simplified version may even have a benefit beyond the learning period, because it has been found in studies of physics problem solving that experts make greater use than do novices of qualitative models in the initial approach to conceptualization. This result may in part be a pedagogical artifact, but presumably a part of expertise is knowing when and how to suppress detail and focus on what is important.

Why then do we choose for our subtraction representation one that involves extra detail for the beginner? The answer to this question reflects conventional wisdom in the matter, as voiced by Glennan and Callahan [1]. Briefly, the details are included in order to demystify what may otherwise appear to be arbitrary aspects of the algorithm. The simple arrangement allows the tray to serve as a visual version of a *list*. The elements of this list are the numbers of coins in each compartment. Since a number now appears as a list of smaller numbers, instead of just a digit-string, it is possible to show 42 as either

([• • • •][• •]) or ([• • •][• • • • • • • • • • • • •]).

(On the screen the 12 coins are stacked in rows of 5 for easier comprehension.) Thus, the results of regrouping can be accommodated with no extra representational machinery either visually or with respect to the way things are interpreted. We simply make the compartments big enough to hold more than 9 coins.

Contrast this situation to the case of pencil and paper. Just when the important idea of regrouping is to be conveyed, the pencil and paper student is confronted with new conventions demanding the crossing off of digits and the writing of digits in strange new places. To regroup 42, for example, one crosses out the 4 and writes a 3 and a 1 in slightly different positions within their relative columns and with different kinds of interpretations. At this point, moreover, the 3, 1, and 2 are all active, with no clear boundary between any two of them, and yet they do not collectively mean 312.

Transfer Via Numberland

Transferability concerns the question of how smoothly the student will be able to move from capabilities acquired in the computer-based instructional environment to target capabilities in more conventional venues and representations. As noted earlier, transfer need not be automatic; for some learners with some techniques, it must be explicitly learned. Even if a new environment is superior to a traditional method, its value may be undermined or partially offset by difficulty or failure when it comes to transfer.

To facilitate transfer, we have developed a second, alternative screen, called Numberland, that is intermediate in appearance between the Coinland screen and the look of pencil-and-paper subtraction, and that permits solution of the same subtraction problems. Once the student has succeeded at the Coinland version of a problem, the system presents that same problem in Numberland. The student can then use her own valid knowledge of how to proceed in Coinland to instruct herself in Numberland, by toggling back and forth between the two representations, solving the problem in both, more or less in parallel. This capability is implemented for subtraction; ideally, learning the translation between the two screens would begin in the simpler reals of counting, regrouping, and addition, so that its occurrence with subtraction would constitute a continuation.

Numberland looks much like Coinland, though there are some crucial differences. Figure 2 shows the Numberland screen for the same problem as Figure 1, at the same stage of solution (even though the "11" here has no counterpart in Figure 1, a point we will return to). The most noticeable difference is that in Numberland all the coins give way to numbers. For example, the numeral "8" appears, in Figure 2, corresponding to the eight dollar coins in a cup in Figure 1. The three keys for picking up and dropping coins are still active, as are the arrow and ENTER keys. Moreover, the same sequence of keystrokes that solved a problem in Coinland will also solve it in Numberland. This is because at an underlying conceptual level, the two "lands" are identical. Not only is Numberland internally unified with Coinland but, equally important, it is externally unified with the world of paper and pencil, as will be spelled out momentarily. It is this twofold linkage that suits it for the role of fostering transfer.

Further description of Numberland will reveal its external similarity to paper and pencil. In particular, the original problem statement is *not* destroyed by student actions here. The actions that in Coinland move five out of the eight dollars from the take-in tray down to the give-back (answer) tray now cause a "5" to appear down in the answer. For the duration of those actions, the "8" is still present, but it turns white, like the dis-

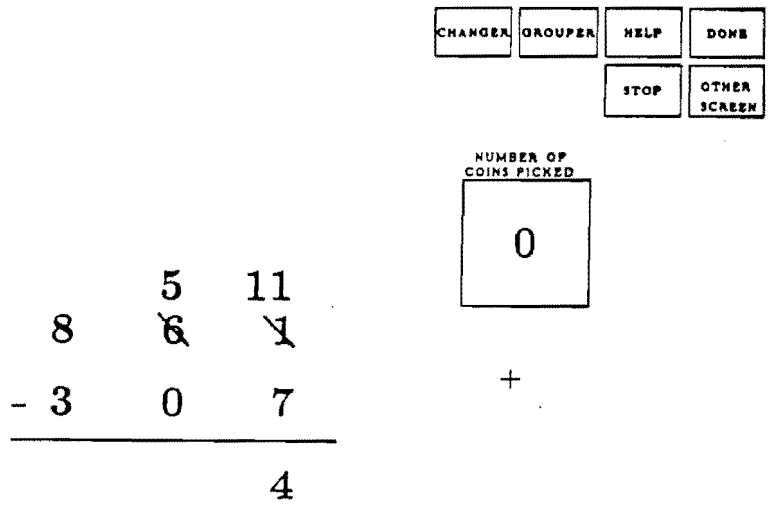

Figure 2 The Numberland screen during solution of the same problem, 861 − 307, as in Figure 1, at the same point in the process. The same keystrokes have occurred as in Figure 1. Conceptually, there are only 7 pennies left in the upper row, even though the number 11 is apparently active there. Therefore, a student who attempts to pick up more than 7 pennies at this point is thwarted, and there is an offer of help.

turbed coins in Coinland, and regains its color at the end. Thus the "8" ends up looking as if nothing had happened, which does not correspond to Coinland (where eight pennies would be reduced to three) but does correspond to the task on paper. A further partial correspondence to the world of paper and pencil is the appearance of the effects of using the changer. In Figure 2, the learner has executed the keystrokes that in Coinland exchanged a dime for pennies in the take-in tray. In Numberland the result is that the minuend, which is 861 or (8 6 1) in list form, becomes (8 5 11) in list form. Notice that the "6" and the "1" are still visible, to provide continuity, but each is now overstruck with a slash. Of course in the figure, the "11" does not mean that there are 11 pennies still in the take-in tray, because four have been lowered so that only seven remain. A learner who forgets this and tries to pick up more than seven pennies is notified why that is not a possible move. The "11" does reliably signify that there are 11 pennies altogether, because we do not allow coin exchange in the give-back gray, in further correspondence to the conventions of paper and pencil.

One last potential contribution to transfer is the slotted, or partially covered, cup. It is harder to extract coins from a slotted cup than from the open ones seen up to now. Slotted cups complicate the environment but will simplify the search. They are to be introduced after the learner has already become familiar with the environment. Recall the observation that a small search space should help reduce inefficient and possibly unenlightening search. The slotted cup would be the external representation of a tightening of the search space. It represents a disincentive, which may be regarded as a variable constraint, in contrast to an absolute one. Specifically, we attach a disincentive to the action of taking coins out of the compartments of the give-back (answer) tray. Even in the absence of such extractions, there would still be enough flexibility to allow an interesting search. The rationale for discouraging that kind of move is that it is invariably evidence of an inefficiency, caused either by the move itself or by an earlier move.

More to the point in this discussion of transfer, altering the answer (the pencil and paper analog of removing coins from the answer) is not permitted in the subtraction algorithm that is taught in most U.S. schools. With that algorithm as the target skill, this variable constraint is a bias toward successful transfer.

To communicate the disincentive visually, the give-back compartments, which otherwise appear as cups without tops, would be drawn with partial tops: each top appearing as a line with a gap, to indicate the side view of a slot. The story is now that it is easy to drop coins in through the slot but hard to get them out once they are in, since the hand that carries coins does not readily fit through the slot. The game score reflects this difficulty by penalizing extractions from slotted cups. Notice that whereas real physical impediments might frustrate a student, the graphical impediments are easily overcome, yet continually provide a reminder and challenge.

Heuristic Guidance

Heuristic guidance (HG) of a student's exploration in problem solving is complementary to semantically constrained exploration (SCE). HG shares SCE's general objectives and is envisaged for use with it, providing guidance within the constraints. HG differs from SCE by being responsive to the choices, especially the most recent choice, that the current student has made in attempting to solve the current problem. Also, the suggestions in HG need not be heeded, unlike constraints. The first subsection places HG in context and presents the basic idea. We then begin to assess its potential for success outside subtraction by looking at some domain characteristics. Finally, we provide the specifics of generating warnings and hints, in terms of a state-based analysis of a subtraction scenario.

The Context and the Basic Idea

In HG, an electronic tutor uses a heuristic function with both cognitive and problem-oriented components to guide exploration. Since such guidance might otherwise be based on a model of the student, formed through a process of diagnosis, HG can be seen as a strategy for reducing the need for diagnosis. It is also quite possible to use HG and student modeling techniques in tandem. In any case, reduced dependence on diagnosis could be highly beneficial, because diagnosis is impossible to do perfectly and difficult to do well, that is, with great detail and reliability. Even for subtraction, it is complex [16]. We shall try to show that, at least in subtraction, HG can generate the kind of advice that is needed for successful learning. After an introductory subsection, we explore limitations on the applicability of the approach and then provide a detailed example of how warnings and hints can be generated.

As envisioned so far, HG can take the form of warnings and hints. When the HG tutor issues cautionary advice to the student, its reasoning, though not necessarily its actual output, goes something like this: "I don't know why you made that move, but it doesn't achieve any obvious improvement and doesn't appear to make things look simpler for you." This sort of warning does not depend on hypothesizing a student model, say, by ascribing a partially deformed version of a standard algorithm. Instead, the method requires the ability to evaluate the new state that results from the student's current move, in comparison with the evaluation of the previous state. We use the word "state" here to refer to a complete specification of all the significant aspects of the

current situation, such as the number of coins of each type in each tray. In a more positive vein, in contrast to issuing a warning, the evaluation of states can also provide the basis for a hint, suggesting a move to a specific alternative state.

To be selective about when to give warnings, an HG electronic tutor needs a heuristic function with both problem-oriented and cognitive components. From a purely problem-solving viewpoint, a heuristic evaluation function may take into account how similar a state is to a goal state (solution) with respect to various attributes. Such information concerns the problem, not the solver. It ignores cognitive limitations and possible strategies to deal with them. Two other classes of candidate traits for desirable states are more explicitly cognitive in nature, namely, the various aspects of simplicity and prominence. A prominent state is, more or less by definition, easy to keep in mind, so it may be looked to as a subgoal. Simple states, by allowing low cognitive load, may make reasoning easier.

How to combine these various characteristics into a single value for the heuristic evaluation function is another aspect of tutorial strategy. One reasonable approach is to add within category and then combine the results disjunctively, by taking their maximum. Using a function formed in this way, a tutor can leave a student alone as long as he or she either is getting closer to a solution state or appears to be successfully pursuing simplicity or prominence.

Domain Characteristics

Not just any evaluation function will be good for HG, and some domains other than subtraction may not have any good ones. Two characteristics of the subtraction domain seem to help make it yield a suitable heuristic function. One is relative freedom of ordering in the move sequence (to be discussed shortly), and the other is a high degree of informational independence. Both of these characteristics show up in the fact that the desirability of ever borrowing (anywhere in any sequence of steps) into a particular column can often be confirmed or denied entirely from information local to the column itself. Relatively free ordering means that there are many equivalent paths, so it severely undermines the possibility of diagnosing in the sense of determining the intended path. HG, however, deals with states, not paths—neither a path chosen by a resident expert nor one attributed to a student.

A somewhat unorthodox approach to teaching subtraction is implicit in this discussion and is in fact adopted in Coinland: no preferred sequence of operations is enforced. For example, in any problem without borrowing, the individual intra-column (digit-wise) subtractions can occur in any order, unlike algorithms commonly taught that prescribe a right-to-left sequence. This neutral stance, which is a natural consequence of the design, is compatible with Self's third slogan: "Empathise with the student's beliefs; don't label them as bugs" [17].

Since order is not imposed, it becomes possible for a student to discover a benefit of (the standard) right-to-left order in some circumstances. If she prematurely shifts all the dimes to the answer, say, but subsequently needs a dime to exchange for pennies, she will have to (at least partially) undo that shift. She may notice this, react appropriately, and ultimately reach the goal state, though not via a minimal length path. Our current (non-HG) system-resident expert can readily detect that its own optimal solution is shorter and can inform the student that the "expert" finished the problem in fewer moves. Somewhat better are the slotted cups referred to above, which warn of inefficiency before the end of a problem. However, the earliest warning of all can be provided

by HG, which, as seen below, will discern when a move first commits the student to a suboptimal path.

From this discussion of alternative paths, it may already be clear that there are many correct algorithms for subtraction. Various ones are taught around the world, so it is not possible for everybody to have the same one. That being the case, we may relax about letting children invent their own if they can. To stress the existence of variety, we express, in terms of Coinland moves, a little known but simple and correct algorithm that we call "prepare-and-do." It has the unusual virtue that it covers all cases, including borrowing across zero, with an unvarying sequence of steps: (1) go through the take-in tray from left to right, using the changer at every position; (2) perform all intracolumn differences (in arbitrary order) and (3) go through the give-back tray, from right to left, grouping wherever possible. Coinland does not, however, support all possible algorithms. One that it does not support in any obvious direct way is one in which borrowing is accomplished by incrementing the subtrahend, because that method violates conservation of quantity; see [18].

This discussion has been based on what we have asserted is the relative looseness in the ordering of steps to solution in Coinland subtraction problems. To be a little more precise about it, moves in this problem space are partially commutative, in the following sense. For any two operations, A and B, such that it is possible to apply them successively in either order to some state, the order is immaterial. That is, if both A(B(s)) and B(A(s)) exist, they are equal. This makes for a space with many cycles and many options at a state. Most to the point, suppose that a student's method dictates the setting of a particular subgoal and that a particular state corresponds to that subgoal. Then there will typically be several equally good paths to that state from the current state and more than one possible first step. Thus, it is not unusual for the set of possible first steps for one approach to overlap those of another approach, making the steps in the overlap ambiguous clues for determining what algorithm, if any, the student is using. Although over the course of time the system might gather many clues in hopes of resolving ambiguity, the student might change (learn) in that same period.

To this flexible domain HG brings flexible tutoring by (1) granting validity to more than one algorithm and (2) allowing exploratory deviation from an algorithm. In contrast to HG, one can envision a tutor with flexibility only on point (2). Such a tutor might allow a student to take more than one step off the standard path but guide the too-deviant student back onto the track. Note, however, that determining how far someone has deviated is not just a simple matter of keeping track of how long ago he or she left the designated path, since despite a long excursion the student may now be almost back on track. This point is especially relevant in a domain with flexible move ordering.

Before leaving this discussion of HG, we put it in some context by mentioning two interesting alternative strategies, each of which renders diagnosis simpler, rather than avoiding it. Singley [19] makes the student post goals, which is like partial self-diagnosis in that it turns over to the student some of the responsibility for figuring out what he or she is up to. In Anderson's Lisp tutor [20], a complex intelligent tutoring system, the price of simple diagnosis is the use of tight constraints on exploration, a tack that, in its severest form, would reduce exploration to mere tracking, with neither of the two types of flexibility described above. On the potentially negative side for HG, it may be subject to the weaknesses of the "weak" method of search, notably inefficiency. However, if it is true that there are pedagogical advantages to allowing exploration, and if we agree that we want to get them, it seems that we have already bought into the idea of search. It is our endeavor first to constrain that search, as we have done by introducing the

Coinland environment, and then to guide it, as we are now suggesting here by introducing the use of a heuristic evaluation function.

To sum up, we have argued that a substantial degree of order-independence in a domain tends to make diagnosis hard, since a step doesn't reveal an algorithm, but that same property tends to make HG, with its state-based evaluation functions, a reasonable alternative for guiding the tutorial process. If this is so, the two techniques are complementary, and each should be used where it is suitable.

Hints and Warnings for Subtraction

To devise a system of HG for a domain, one must first decide what counts as a state and what are the significant attributes of states. For the Coinland version of the subtraction domain, we abstract a state space that consists only of situations that can exist immediately after the completion of a shift of coins or a conversion between denominations. We thereby exclude from statehood those situations that arise from just picking up or unpicking coins and/or moving the cursor. Although we have earlier justified having the learner pick up one coin at a time, those detailed actions have little or no significance for solving the problem at hand and indeed have no analog in the world of pencil and paper. From this viewpoint, picking up coins is a micro-operator used to form operands for the principal operators of shifting and exchange.

Obvious candidates for significant attributes of a state are the numbers of coins in various compartments. We shall use all six of these numbers. Less obvious are attributes that summarize the problem-solving history, like the net number of dimes converted to pennies (changer uses minus grouper uses) and net number of dollars converted to dimes. We shall use both. These last two numbers are not visible on the screen so they are less appropriate for use in explanation, but they can help decide whether to issue a warning, and they are useful in generating unexplained hints. (The number of coins picked up is not a useful attribute, since it is always zero in the situations allowed as states.) These eight state attributes are not all independent, since for example, the combined pennies in the two trays, minus 10 times the net number of dime-for-pennies exchanges, is invariant. Two other similar constraints hold, so that three of our attributes could be dropped without loss of information, though with some loss of simplicity in the discussion below.

From a purely problem-solving viewpoint, the most desirable value for a compartment of the take-in tray is its target value, the corresponding digit of the price, because this is its correct final value in any solution. Of course the correct final values for the give-back compartments are also desirable values for those attributes, but they are not given in the problem statement, as the price digits are. Therefore these values, like the net conversion numbers, are used only to formulate unexplained hints.

We now work through a scenario with a specific problem, 503 − 7, showing how warnings and hints come into play. In accord with the discussion above, the initial state in denoted 5,0,3;0,0,0;0,0. The first six numbers indicate how many coins are in the compartments of the two trays, and the last two are the net number of exchanges of, respectively, dollars for dimes and dimes for pennies. The correct final state is 0,0,7;4,9,6;1,1. Suppose the student begins by lowering all the dollars to get 0,0,3;5,0,0;0,0 and then exchanges a dollar within the given-back tray to get 0,0,3;4,10,0;1,0.

In the pencil-and-paper world, corresponding operations are incorrect with respect to the standard algorithm, and indeed the second one would require an extension of the

usual representation. Nevertheless, they do appear to have some possible merit in terms of our attributes, so no warning is given. In particular, the first move meets a take-in tray objective of being equal to the implicit price digit of zero in the hundreds column. Introducing a zero is also regarded as a potential cognitive benefit of simplification. The second move in the scenario brings both the give-back dollars and the net number of dollar exchanges to their correct final values. Note that exchanges are counted without regard to the tray in which they are performed. Strange as these moves may seem, it may indeed turn out to be wise to have tolerated them, since they do flow from the not incorrect solution strategy, "give enough dollars back to the customer so that it is the customer who must make the change."

Converting another dollar at this point does deserve a warning. That move yields the state 0,0,3;3,20,0;2,0, thereby disturbing some goal values, with no compensating benefits. To generate a hint, we use an algorithm that loops through the columns from right to left. In each column it first tries to achieve what we will call G1, the visible objective of equating the take-in compartment to the price digit, by means of a shift in one direction or the other. If G1 is not already satisfied, it can be achieved provided the total number of coins of the current denomination is greater than the price digit, but by an amount no more than 20 (the capacity of a compartment). For example, if a (confused) student has 14 pennies in each tray while solving a problem in which the price is 5, the system should not suggest lowering 9 pennies to get the required 5 above but an impossible 23 below. In the present example, with only 3 pennies present altogether, there is no shift that can get the take-in tray to have 7.

If G1 is already satisfied or cannot be satisfied by a legal shift, the hint generator instead seeks a hint for achieving G2, getting closer to the correct net number of exchanges. Preference is given for an exchange within the take-in tray. In the present example, the target number is 1 and the current number is 0, so the hint would be to get ten pennies for a dime by using the changer. Since there is no dime in the take-in tray, the hint here would be to change a dime from the give-back tray. Suppose that the student accepts this hint, so the state become 0,0,3;3,19,10;2,1. If a hint were requested again, from this state, it would be to move 4 pennies up to the take-in tray, giving that tray 7 pennies, its target number, and making the new state 0,0,7;3,19,6;2,1.

If yet another hint is requested, the hint generator algorithm will find that G1 and G2 are satisfied for pennies, so it will move to a consideration of dimes. There, G1 is satisfied but G2 is not. Since there has been an excess of changer uses, 2 instead of the 1 called for, the hint is to use the grouper. Since 10 dimes are not to be found in the take-in tray, the hint is to group 10 dimes from the give-back tray, thereby solving the problem. Interestingly, this move happens to be the inverse of an earlier (poor) move. There is a rationale for this move that can be expressed in terms of the objectives of Coinland subtraction and can therefore be cited when communicating the hint to the student. This is the requirement that, in the final state for every problem, no compartment in the give-back tray may have more than 9 coins.

Before leaving this discussion of HG for subtraction, we comment on its use with two particular algorithms, first Ikeda's algorithm [21] and then the standard one. The advantage of Ikeda's method lies in its intriguing property of not requiring the student to learn the harder subtraction facts like $12 - 7$ and $17 - 9$, in which the minuend is greater than 9. To solve $62 - 37$, the student in effect regroups 62 not into 50 and 12 but rather into 52 and 10, then takes 7 from the 10, and to the resulting 3 adds the 2 taken out of 52 to get 5 in the ones column of the answer, leaving 50 from which to take 30 to get a 2 in the tens column of the answer.

HC accommodates Ikeda's method without having been explicitly designed for it, as will be seen momentarily. When subtracting 62 − 37, Ikeda's method corresponds to lowering 2 pennies, then regrouping, then lowering 3 more, then finishing off by lowering 2 dimes. The first move scores well because it creates an empty compartment, regarded as an aspect of simplicity or relatively low cognitive load. Note that it would be difficult if not impossible to infer, over several problems, with a possibly changing student, whether this kind of move is really being used systematically and therefore deserves to be credited with possibly being part of an algorithm, or is just a part of some haphazard searching.

Finally, suppose one wishes to push the student toward move sequences compatible with the standard algorithm. One can do this with an "advanced" version of the game, formed by introducing the extra constraint that coins may not be taken out of the give-back tray, even for a penalty in the score. In this version, it makes sense to warn against certain moves in our example scenario that did not receive warnings in the above discussion. In particular, the 9-coin limit on give-back compartments applies to all states, via an explainable warning that says, in effect "That's too many dimes (or pennies) in the give-back tray for a final answer, and you are not allowed to take them out, so you better not put them in." It is worth noting that this constraint is not too terribly confining, as can be seen from the fact that it permits both Ikeda's method and prepare-and-do (see above), in addition to the standard algorithm.

The Syllabus

Both SCE and HG have their effects within a single problem, as the learner seeks a solution to it. Up a level and outside the individual problem is the broader issue of how to choose a particular problem, among the huge number available, to meet the need to work on a particular topic in a syllabus. This matter in turn raises the issue of what material goes into a syllabus and how to structure it for orderly coverage. The first subsection deals with syllabus issues with extended attention to subtraction. Then we turn to navigating that syllabus, that is, to selecting a topic and problem. The final subsection concerns the nature of subtraction.

The syllabus not only specifies material but also makes commitments to the granularity of its organization. For example, given that one-digit addition with one-digit answers is a topic, should adding 1 be split off as an introductory subtopic? The level of granularity to which the topics are dissected should be coarse enough for the units being learned to be coherent, yet fine enough to permit the explicit expression of constraints and preferences with respect to the order of presentation of material. These constraints and preferences structure the material and thereby determine in what order(s) it is to be learned. One important kind of constraint is the prerequisite relation, imposed between two topics or concepts. One topic is an absolute prerequisite of another if failure to master the first completely undermines the learning of the second. A variable strength version is also possible. Note that a syllabus closely resembles what is commonly called *domain expertise*, but it emphasizes declarative principles over compiled procedures. The earlier emphasis on conservation of quantity exemplifies the distinction and is in this regard indebted to insights that drove the evolution of Guidon II [22] away from unexamined expert performance rules.

The Subtraction Syllabus

Turning now to the specifics of subtraction, note first that there are roughly half a million ways to choose a pair of three-digit numbers for a subtraction problem with a positive answer (and about 50 million with four-digit numbers), but many of these are essentially equivalent in terms of cognitive demands. To carve up the space of possible problems into structured topics for a syllabus, we will appeal to the notion of subgoal and to other constructs from the study of artificial intelligence (AI), thereby pursuing a strategy that has potential use in domains outside arithmetic.

First, consider the requisite values of parameters in the goal state. To complete a Coinland subtraction problem successfully, one must get each compartment in the take-in tray to have a number of coins equal to its corresponding target digit. This requirement gives rise to three subgoals, one for each compartment or digit. These subgoals categorize problems according to whether or not a given subgoal is already satisfied in the starting configuration. This is equivalent for a subtraction problem to whether the top and bottom digit in a column are the same. Problems can be further divided according to whether in the starting configuration some subgoal can be achieved in a single move, as when the top digit in a column exceeds the bottom one. Two additional subgoals arise from the requirement that the dime and penny compartments in the give-back tray must each have no more than nine coins to represent a permissible answer.

Turning from subgoals to moves, we note that some starting configurations are of special interest for a different kind of reason, namely that they make certain initial moves impossible. For example, in the problem 100 − 7, the two initially empty compartments in the take-in tray constrain the learner to just two possibilities for the first move, specifically, exchanging the dollar for ten dimes or moving the dollar to the answer tray. This paucity of options is attributable to the semantic constraints of the Coinland environment. In this way, we hope and expect that Coinland will facilitate solution in the foregoing problem with its renowned pitfall of borrowing across zero. If that turned out to be true and if the success in Coinland were to transfer, it would strongly suggest that the original difficulty was an education artifact.

The foregoing example has the potential for a different pitfall. To follow the usual algorithm, one must undo satisfaction of an initially satisfied subgoal, specifically that of having the number of dimes in the take-in tray equal to zero, the target digit in the middle column. To preserve satisfaction of that subgoal, one could move the dollar down, make change to get dimes and then pennies in the give-back tray, and raise seven pennies to the take-in tray. This approach answers the question of what is left when 7 is taken from 100, rather than how much must be taken from 100 in order to leave 7 behind. Volpel [23] discusses such ways of thinking about subtraction.

There are two kinds of moves in Coinland, the shifting and the exchange of coins. Shifting is simpler in that it involves only one denomination at a time and conserves physical (graphical) objects, as well as quantity. Moreover, shifting is used in all problems (except those whose answer is zero). These considerations suggest that borrowing, which is equivalent to exchange (to a lower denomination), is the more crucial type of move and so should be considered at the top level in our taxonomy of problems. Of course, one already knows that borrowing is crucial; what is of interest is that this conclusion has flowed out of a general consideration of exploratory problem-solving moves.

In summary to this point, we have considered two aspects of problem-solving: move types and sub-goals. Consideration of move types has suggested the importance of borrowing, and our earlier discussion of subgoals showed that the relative sizes of zero, T and B are important, where T and B are, respectively, the top and bottom digits in a column. There are six distinct "B-T-zero patterns," as follows:

$$0 = B = T$$
$$0 = B < T$$
$$0 = T < B$$
$$0 < B = T$$
$$0 < B < T$$
$$0 < T < B$$

These distinctions yield 92 categories of subtraction problems, each with a three-digit minuend and a non-negative result. Most of these, 72 ($= 6 \times 6 \times 2$), arise because any of the above six orderings can hold in the ones and tens columns, along with either of the two orderings for which $T > B$ in the hundreds column. Only another 20 possibilities arise with $0 < T = B$ in the hundreds column, since the other two columns must then be consistent with a non-negative result. Moving to four-digit problems, for which there is some pedagogical justification, increases the number of categories roughly six fold. This is better than going up by a factor of a hundred, as is approximately the case for possible individual problems, but still of finer grain than is convenient or necessary for tutorial purposes. A coarser categorization is available at the level of borrowing patterns, which arise above from considering move types. Distinguishing three-digit problems according to borrowing, seven patterns arise, three of them having at most a single borrow:

NB: No Borrow
$T \geq B$ in each column.

B-ONES: Borrow only into the ONES column
$T < B$ for 1s; $T > B$ for 10s; $T \geq B$ for 100s.

B-TENS: Borrow only into the TENS column
$T \geq B$ for 1s; $T < B$ for 10s; $T > B$ for 100s.

The remaining four patterns all involve borrowing in both columns, so $T < B$ for 1s, $T \leq B$ for 10s, and $T > B$ for 100s. Various special conditions are indicated for 10s, according to whether or not $0 = T$ and whether or not $T = B$.

B-TWICE; Borrow TWICE, but with none of the special conditions below, so $0 < T < B$ for 10s.

BAZ: Borrow Across Zero.
$0 = T < B$ for 10s.

UNDO: UNDO the initially met subgoal.
$0 < T = B$ for 10s.

BAZ-UNDO: The equalities in BAZ and UNDO hold.
$0 = T = B$ for 10s.

The last few distinctions are motivated not only by subgoal considerations but also by another general AI problem-solving notion, that of constraints on the order of moves. In particular it is the borrowing (exchange) moves that are constrained. In BAZ, borrowing into the tens column necessarily precedes borrowing into the ones column. In UNDO, the reverse order is strongly suggested, since with B = T in the tens column, there is initially no apparent need to borrow into that column. BAZ-UNDO provides a basis for both the preceding arguments with their associated opposite orderings for borrowing. Confusion could arise if BAZ and UNDO were each encoded by a learner as a slightly over-general rule with antecedents of, respectively, T = 0, and T = B. Then both rules would be applicable under the conditions of BAZ-UNDO, giving conflicting advice.

One important reason for creating this taxonomy is to impose a partial order on the material by the prerequisite relations, depicted in Figure 3. The easiest pattern is at the bottom of the figure, the hardest at the top. Although B-ONES and B-TENS are shown as unordered with respect to each other, there is a rationale other than the prerequisite structure for presenting B-ONES first. Since it can be taught with only two denominations present, it will be taught before B-TENS if the tutor has a principle of using as simple a problem as possible when introducing a new problem characteristic.

There are at least five discernible granularity levels in an over-all taxonomy of subtraction problems. Beneath the top level (at which no distinction are made) is the level of borrowing patterns, then that of the B-T-zero patterns. A fourth level concerns distinctions between relatively difficult facts like 17 − 9 = 8 and easier ones like 12 − 6 = 6 (discussed below). Fifth and last is the level of individual problems. One might insert yet another level, concerning number of digits, right beneath the top level. Viewing subtraction within arithmetic brings the number of levels to seven.

Selecting a Topic and a Problem

Having structured the material, we turn to the question of navigating within that structure. Paying attention to the individual learner and responding appropriately could begin with such high-level decisions as the choice of subject matter and instructional technique. Having chosen subtraction as the subject matter, the tutor may encounter a student

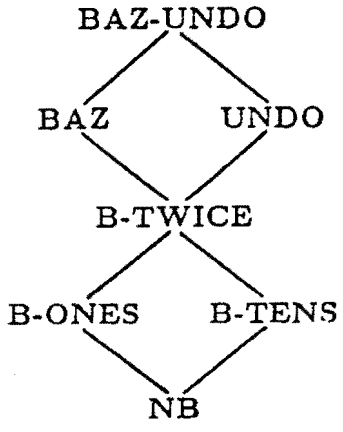

Figure 3 A coarse-grained taxonomy and prerequisite structure for three-digit subtraction. The nodes are categories of problems, defined in the text. A problem type at the lower end of an arc should be mastered before the student attempts the type at the upper end of that arc.

who needs to be backed up to its prerequisites. We have tentatively dealt with this problem by some introductory exercises on the prerequisites. We will discuss some general ideas on prerequisites below. When the mode of interaction is problem-solving, the tutor must be able to select suitable problems and, within a problem, know when to interrupt and what to say when doing so. We considered the latter problem earlier under the rubric of HG. We now consider how to choose a problem, in two steps. First, determine whether to continue on the same (sub)topic within the syllabus, and know how to choose a new one when appropriate. Second, choose a good problem on the topic.

Given the kind of taxonomy described in the syllabus section, be it in subtraction or some other subject, how does the tutorial component navigate it? We now address this question of the choice of topic within any syllabus. The areas or topics in a syllabus may have various individual attributes and pairwise relations that are tutorially significant. These attributes and relationships are the key to determining a good choice for the next topic. Useful attributes of a topic are its importance, its degree of difficulty, and the extent to which the current student has apparently mastered it. One relation that would seem indispensable is the notion of a prerequisite. Part-whole relations may also prove helpful and possibly too the notion that one topic should immediately precede another. Some of this information, like the prerequisite relation, may be regarded as imposing firm constraints on the ordering of subject matter, while others, like difficulty, might indicate only a preference. Notice that predecessor implies prerequisite and that prerequisite must be acyclic if it is to be enforced.

If the foregoing kinds of information can be supplied by a domain expert, it is possible to assemble automatically from that information an order in which topics occur. Suppose that topic A is a prerequisite of B and B is a prerequisite for C. Let it also be specified that A should be an immediate predecessor of C and further that C is very important. We will first consider how a planner might proceed and then look at a myopic forward reasoner. To formulate an order among the topics, one might first consider the important topic, C, and put it into the plan. If the predecessor relation is taken as a firm constraint, then the new plan becomes ((A C)), where the outer list can tolerate insertions anywhere, but the inner list only at its ends. Neither permits reordering. Checking prerequisites, we must add B before C. We amend the plan to (B(A C)) which is less constrained than ((B A C)) and yet meets the requirements considered so far. Another check of prerequisites leads to the final plan (A B(A C)).

Now consider a forward reasoning process. C is rejected as the initial topic because its prerequisite and predecessor have not been studied. B also has a prerequisite, so A becomes the first topic. When A is finished, C, for which it is predecessor, is considered, but the prerequisites of C are not met, so B becomes the next topic. After that, only C remains unmastered, but it cannot be taken up unless A immediately precedes it. If the predecessor constraint is again regarded as binding, then A is repeated, after which the conditions for studying C are met. The sequence is the same as that found earlier by the planner.

The use of problem solving requires that the tutor select an appropriate new problem from time to time. In our earlier structuring of the subtraction syllabus, the topics were defined in terms of relative size of digit values, so once a topic is chosen, finding a problem is a matter of generating digits that satisfy those relationships. The choice of particular digits influences the difficulty level of a problem within the particular topic. Partly on the basis of findings by Clapp [24], we assume certain generalizations about the relationships between numerical properties and the resulting difficulty. Actual use of the system will enable us to check these assumptions. For example, we assume that the

difficulty of a one-digit subtraction fact increases monotonically with the minimum of the subtrahend and result, for a given minuend, making 9-7 harder than 9-8. Getting a problem at the right level of difficulty within a topic can be achieved with a repertoire of three methods, one to generate an easy problem and two others for transforming a problem into one that is harder if the student is doing well, or easier otherwise.

The Nature of the Material

In addition to breaking topics into partially ordered subtopics, there must be a decision about the nature of the material. Should an arithmetic operation like multidigit subtraction be regarded, and also taught, as something that is like production rule programming, procedural programming, or something else? In other words, in what form, possibly a computational paradigm, shall we express the material to be learned? Presumably one wants to choose the form in which the material will be most readily learnable or "communicable" (in the sense of Wenger [18]), best understood and easiest to use. Learnability and usability in turn depend on fit with cognitive capabilities. New material fits to the extent that its parts are equal to or closely analogous to what is already known or can easily be figured out by types of reasoning that are relatively effortless for a person.

Subtraction is a good domain in which to examine the choice of expressive form. It is typically taught as an algorithm but can also be expressed as a set of rules or even as a set of principles. We next spell out these three possible viewpoints for subtraction, commenting also on the nature of the comparison relations, greater/less than. After that we comment on the facts or data used in conjunction with the algorithms, rules, or principles.

First, as an algorithm, subtraction of multidigit numbers can be expressed as an outer loop over the columns from right to left. For each column, there is an if-then structure that branches according to whether borrowing is needed. Borrowing itself involves two loops, one to find the nearest leftward non-zero digit in the minuend, and one to carry out the required succession of regroupings (possibly only one regrouping). One can also treat comparison as an algorithm, such as the following: If one number is longer it is larger; for numbers of equal length, start at the leftmost column and search rightward for an instance of unequal digits in corresponding columns; etc.

Notice that the search loop in this comparison algorithm and the main processing loop in the subtraction algorithm move in opposite directions. For this reason, their joint use might tend to obscure the similarity of the concepts they embody. After all, comparison is the idea of one number being larger than another, whereas subtraction reveals by how much. In the Coinland environment it is possible to emphasize this similarity by creating a comparison exercise directly analogous to the subtraction exercise. Specifically, the student can be given two trays to compare and allowed to make value-preserving exchanges in either until one has at least as many coins of each denomination and more of at least one denomination.

As a set of rules, subtraction can take the form provided in [25]. For example, as adapted in [18], the rule for processing a column that has no digit in the subtrahend is:

IF number-a is in column-a and row-a
  AND row-a is above row-b
  AND your are processing column-a
  AND there is no result for column-a

AND you are focused on column-a
AND there is no number in column-a and row-b
THEN write number-a as the result for column-a.

One benefit of rules like these is that they break up the requisite knowledge into chunks that are mutually independent and are of a suitable size for learning. A related aspect that proponents of rules appreciate is that they can be individually deformed by small perturbations in their conditions, given in the "if" part of the rule, on when to apply the operation specified in the "then" part. For example, one of the conjuncts might be omitted. In this way the system can generate malrules or bugs that may provide an account of a given student's incorrect performance.

Without the explicit sequencing that procedures impose, rules need extra conditions to keep from being used at inappropriate times. These conditions would always be satisfied (and not need to be checked) at the right position in the ordering of a procedure. Using excessive rule conditions in this way violates the dictum that one should use a rule system only for processing that is inherently unordered. An apparent compensation for the excessive use of explicit conditions is that the explicit loops found in the algorithm become unnecessary. However, there must be looping in the control structure of the system that executes the production rules.

Coinland, in contrast to the six-condition rule given above, places no firm conditions on the applicability of operations. Its operations of shifting and exchanging are applicable at all times, in the sense that they never violate the semantics of the domain. A learner may have inappropriate conditions for the use of these operations, but that is a matter of inefficiency rather than incorrectness, a weaker strain of bug. By moving subtraction into a general problem-solving framework, the Coinland approach generalizes the problem of seeking the conditions that favor the use of an operation. Here an operation is to be used when it appears to move toward a goal state, perhaps by achieving some subgoal or getting to a state with a high heuristic value.

Third and last, having looked at subtraction as an algorithm and as a set of rules, consider what it might mean to regard subtraction as a set of principles. In particular, we will look at a specific (partial) set of subtraction principles. These principles would come later in the syllabus than—and can therefore depend upon—an understanding of small positive integers, counting forward and backward, and the principles of addition. The first principle is just definitional.

1. A subtraction problem includes two numbers, called the minuend and the subtrahend, and demands an answer which is the former reduced by the latter. ("Reduce" is defined in 2.–4.).
2. Reducing a number by 1 yields its predecessor in the counting sequence. (This relies on knowing how to count backward).
3. A reduction may consist of several component reductions by amounts that add up to the (composite) reduction amount.
4. The order among component reductions (as in 3.) is immaterial. (This follows from (3.) and the fact that order is immaterial in addition).

Clearly such principles are to be conveyed far more subtly than simply by stating them. What may be less obvious is that to ascribe knowledge and use of such principles to a child is not to claim that the child can articulate them, any more than we demand that a child state the rules of grammar before we concede that she speaks her native tongue.

Having introduced these sample principles, we are in a position to say something more concrete about the issue raised above, prerequisites and granularity. As for prerequisites, first note that 2. must precede 3., for the latter to make sense. Second, principles 1.–4. do not include place numeration, so it would seem that the principles of place numeration, collectively, do not have a prerequisite relationship to 1.–4. in either direction; that is, place numeration can be learned earlier or later than these principles of subtraction. A child possessing principles 1.–4, and able to use them, could solve subtraction problems with small integers, without having knowledge of place numeration (much less a complete algorithm for subtraction). The principles support the solution of $7 - 2$ by counting backward from 7 to 6 to 5. Having memorized that result, a child might us it to solve $7 - 3$ in just two steps, going from 7 to 5 to 4.

Such a knowledge of subtraction potentially extends in applicability to new numbers as soon as the counting order is learned. For example, without having full command of subtraction a child might nevertheless be able to deduce that "two years ago" in 1989 refers to 1987. Once a further principle concerning the subtraction of 10 is mastered, it becomes conceivable to be able to subtract 11, 12, 21, etc., by principle-based exploration, a possibility for which we believe that Coinland can provide a basis.

One last form of knowledge, simple but essential, is the fact. It is important that when a student is working on a problem in some domain the relevant facts of that domain are indeed facts for that student. If instead they are not facts but actually little problems that the student has to work out, they will presumably thereby interrupt the flow of thought in the proper domain. This discussion hints a kind of "super-prerequisite" relationship, one in which an easier topic should be not just mastered as a procedure but compiled into a set of facts before the related harder topic is taken up. Even in such a situation, however, the facts need not be treated monolithically. That is, not all the super-prerequisite facts of an entire domain need be known before work can begin in that domain. For example, it may be possible to teach multiplication to a student who does not know the whole multiplication table, if problems are restricted to those digitwise products with which the student is familiar.

Future Directions, Testing, Acknowledgments

We have implemented subtraction and a variety of related games designed to foster learning of both the Coinland interface itself and earlier topics in the curriculum, including the notion of a place value system of expressing numerical quantity. We also plan to extend the curriculum upward at least as far as division by small integers. As the curriculum becomes more thoroughly covered, it will be possible to make a more meaningful selection of where within it a particular student ought to be. It will then make sense for us to refine our currently primitive diagnostics and tutoring decisions. The system is written in C and is being developed on a PC/AT, but can be delivered on a PC/XT.

We recently conducted extensive testing in grades 1–3 in a local elementary school and earlier did some pilot testing. Though even the recent tests were informal, we found the results very encouraging. Students found Coinland comprehensible and engaging, and some went on to use it to check or instruct themselves in Numberland. A few ultimately reached an apparent knowledge of subtraction aspects beyond what the teacher believed they knew. Such testing is not only part of the design cycle but will

ultimately serve as evaluation of the system, including transfer to paper-and-pencil arithmetic.

The authors will continue to seek the valuable advice and assistance of Dr. Thomas Nuttall, an education specialist in Fairfax County, VA, who recently concluded three years on the faculty of George Mason University. We gratefully acknowledge Joseph Psotka's insightful comments on an earlier draft of this paper.

References

1. V. J. Glennan and L. G. Callahan, *Elementary school mathematics: A guide to current research (3d ed.)*, Washington, DC: Association for Supervision and Curriculum Development (NEA), 1968.
2. J. D. Holland, E. L. Hutchins, and L. Weitzman, "STEAMER: An interactive, inspectable simulation-based training system," *AI Magazine*, 5:2, 15–27, 1984.
3. H. Hamburger and A. Lodgher, "Exploration in Coinland," *Proceedings of the International Conference on Computer-Assisted Learning*, Dallas, TX, April, 1989, Berlin: Springer-Verlag, 1989.
4. T. P. Carpenter, "Learning to add and subtract: An exercise in problem solving," in E. A. Silver (ed.), *Teaching and learning mathematical problem-solving: Multiple research perspectives*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1985.
5. A. Collins, J. S. Brown, and S. E. Newman, "Cognitive apprenticeship: Teaching the craft of reading, writing and mathematics," in L. B. Resnick (ed.), *Knowing, learning and instruction: Essays in honor of Robert Glaser*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1989.
6. L. B. Resnick and S. Omanson, "Learning to understand arithmetic," in R. Glaser (ed.) *Advances in instructional psychology, vol. 3*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1987.
7. M. Lampert, "Knowing, doing, and teaching multiplication" *Cognition and Instruction*, 3:4, 305–42, 1986.
8. J. S. Brown and R. R. Burton, "Diagnostic models for procedural bugs in basic mathematical skills," *Cognitive Science*, 2, 155–192, 1978.
9. J. S. Brown and K. Van Lehn, "Repair theory: A generative theory of bugs in procedural skills," *Cognitive Science*, 4, 379–426, 1982.
10. S. Papert, *Mindstorms: Children, computers, and powerful ideas*, New York: Basic Books, 1980.
11. H. Hamburger and S. Crain, "Plans and semantics in human processing of language," *Cognitive Science*, 11:1, 1987.
12. L. B. Resnick, "Syntax and semantics in learning to subtract," in T. P. Carpenter, J. Moser, and T. Romberg (eds.), *Addition and subtraction: A cognitive perspective*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.
13. T. W. Malone, "Heuristics for designing enjoyable user interfaces: Lessons from computer games," *Proceedings of the ACM conference on human factors in computer systems*, 1982.
14. CERI (Centre for educational research and innovation), *Information technologies and basic learning: Reading, writing, science and mathematics*, Paris, France: OECD (Organization for Economic Cooperation and Developmnent), 1987.
15. B. Y. White and J. R. Frederiksen, "Qualitative models and intelligent learning environments," in R. Lawler and M. Yazdani (eds.), *Artificial intelligence and education*, Norwood, NJ: Ablex Publishing Co., 1987.
16. R. R. Burton, "Diagnosing bugs in a simple procedural skill," in D. Sleeman and J. S. Brown (eds.), *Intelligent tutoring systems*, New York: Academic Press, 1982.
17. J. A. Self, "Bypassing the intractable problem of student modelling," *Proceedings of the conference on intelligent tutoring systems*, Montreal, 1988.

18. E. Wenger, *Artificial intelligence and tutoring systems*, Los Altos, CA: Morgan Kaufman, 1987.
19. M. K. Singley, "The effect of goal posting on operator selection," *Proceedings of the Third International Conference on Artificial Intelligence and Education*, Pittsburgh, PA, 1987.
20. J. R. Anderson, "Cognitive psychology and intelligent tutoring," *Proceedings of the Cognitive Science Society Conference, Boulder, CO*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1984.
21. H. Ikeda and M. Ando, "A new algorithm for subtraction?" *The Arithmetic Teacher*, 21, 716–719, 1974.
22. W. J. Clancey, *Knowledge-based tutoring: The GUIDON project*, Cambridge, MA: MIT Press, 1987.
23. M. C. Volpel, *Concepts and methods of arithmetic*, New York: Dover Publications, Inc., 1964.
24. F. L. Clapp, *The number combinations, their relative difficulty and the frequency of their appearance in textbooks*, Madison, WI: Bureau of Education Research, University of Wisconsin, 1924.
25. P. Langley, J. Wogulis, and S. Ohlsson, "Rules and principles in cognitive diagnosis," in N. Fredericksen (ed.), *Diagnostic monitoring of skill and knowledge acquisition*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1987.